

HOW TO DEVELOP PEN-CENTRIC APPLICATIONS

THIS ISSUE:



July/August 1994
Display Until 8/31/94

os/2

Developer

THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

WRITING PEN-BASED APPLICATIONS

Buyer's Guides:

- **COMPILERS AND INTERPRETERS**
- **TRAINING**

C++ Class Library Systems

*SOM and Workplace
Shell Programming*

*Pen/Audio
Application Development*

**PEN-BASED
COMPUTING
GETS REAL
ON OS/2**

U.S. \$9.95 Vol. 6 No. 4



0 74470 12531

A Miller Freeman Publication

Replacement

Pen at the Chicago Mercantile Exchange

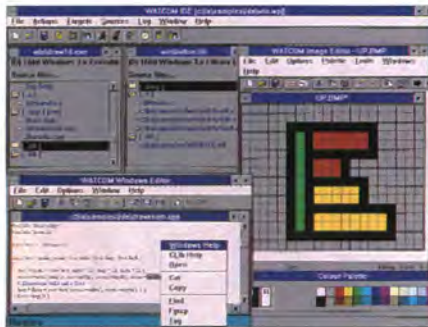
Watcom C/C++ 10.0

ACCELERATE
Your C and C++
Application Development

The new Watcom C/C++ 10.0 development system simplifies and accelerates development of high-performance, multi-platform 16- and 32-bit applications. Watcom C/C++ 10.0 delivers productivity and performance, combining our state-of-the-art compiler technology with a new, integrated development environment (IDE) and comprehensive set of tools.

New Integrated Development Environment and Tools

The new IDE is built to simplify the complexities of real-world application development and make it easy to exploit the high-performance, multi-platform power of Watcom C/C++ 10.0. In a single "project" you can build multiple EXEs, DLLs, and LIBs, targeting several different platforms. The IDE simplifies each stage of development from compiling and linking to debugging and performance tuning. The package includes versions of the IDE and tools for all three host platforms (Windows 3.x, OS/2 2.x and Windows NT).



Watcom C/C++ 10.0 includes a source editor with syntax highlighting, a suite of resource editors, testing and monitoring tools for Windows 3.x and NT development.



The advanced multi-platform debugger accelerates the development cycle by increasing the bandwidth between you and your application.

Multiple Platforms in a Single Package

Watcom C/C++ 10.0 supports development of applications targeting an incredible array of platforms: DOS, Windows 3.x, OS/2 1.x, 32-bit DOS (includes royalty-free DOS extender), OS/2 2.x, Windows NT, Win32s, 32-bit Windows 3.x and Novell NLMs. To maximize the potential on individual platforms, Watcom C/C++ 10.0 extends the capabilities of the core, multi-platform toolset with platform-specific tools, SDKs and libraries. This extensive support is amplified by the cross-platform capabilities of the IDE and tools, which enable building applications for a wide range of target environments from any of the host systems.

The Best Optimization Technology

Watcom C/C++ 10.0 combines both 16- and 32-bit compilers in a single package, providing you with the industry-leading optimizing compiler team. PC Magazine tested performance of industry standard C and C++ compilers and said: "the fastest executables created during testing came from Watcom C/C++³², Version 9.5, while the 16-bit version of the same compiler produced the smallest executables"¹. Now, with Watcom C/C++ 10.0, this competitive advantage is delivered with our easy-to-use development environment and tools.

Watcom C/C++ 10.0 delivers all this in a single package!

- New integrated development environment hosted on Windows, OS/2 and Windows NT
- Comprehensive suite of multi-platform development tools including debugger, browser, profiler and more
- Professional source editor, resource editors, testing and monitoring tools hosted on Windows and Windows NT
- Target Platforms include:
 - 16-bit:** DOS • Windows 3.x • OS/2 1.x
 - 32-bit:** Extended DOS • Windows NT • Win32s • OS/2 2.x • 32-bit Windows 3.x • Novell NLM • AutoCAD ADS/ADI
- Both 16-bit and 32-bit compilers for C and C++, the industry's best code optimizer, faster compile times with pre-compiled headers, C++ supports templates, exception handling and the Microsoft Foundation Class library (MFC)
- Licensed components from:
 - Microsoft Windows 3.1 SDK
 - Microsoft Windows NT SDK
 - Novell NLM SDK v4.0
 - IBM OS/2 Toolkit v2.1
 - Microsoft MFC Class library
- Includes Rational System's DOS/4GW 32-bit DOS extender with royalty-free distribution
- Significantly expanded and revised on-line documentation
- And more!

Suggested Retail Price:

Watcom C/C++ 10.0 CD-ROM Edition
(CD-ROM with on-line documentation) **\$350***

Watcom C/C++ 10.0
(CD-ROM with printed documentation) **\$450***

Upgrades:

(for owners of Watcom C/C++³² or Watcom C/C++¹⁶ v9.5)

Watcom C/C++ 10.0
CD-ROM Upgrade Edition **\$149***

LIMITED TIME \$199*
INTRODUCTORY SPECIAL
Watcom C/C++ 10.0 CD-ROM Edition

1-800-265-4555

Watcom
A Powersoft Company

Watcom International 415 Phillip Street, Waterloo, Ontario, Canada N2L 3X2 Telephone (519) 886-3700 Fax (519) 747-4971

* Price in US dollars. Does not include freight and taxes where applicable. Authorized dealers may sell for less. \$199 Special Offer is available until October 31, 1994. Watcom and the Lightning Device are trademarks of Watcom International Corp. DOS/4GW is a trademark of Rational Systems Inc. Other trademarks are properties of their respective owners. ©Copyright 1994 Watcom International Corp. PC Magazine, March 29, 1994

Circle Reader Service Number 1

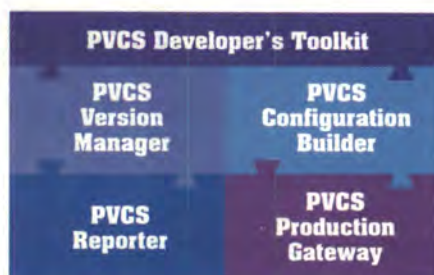


"We chose PVCS because the vendors of our new development tools recommended it. They told us Software Configuration Management was a necessity for productive development on the LAN. I'm glad we listened to them. PVCS has proven to be invaluable. We now have superb control over all development efforts."

The Major Development Tool Vendors Agree: PVCS Is Essential for Productive Development on the LAN

Selected Vendors:

Borland International IDE
Digitalk Team/V
Gupta Technologies, Inc.
HP SoftBench
IBM SDE/6000
IBM OS/2
Interactive Unix
Micro Focus COBOL
Microsoft Visual C++
Microsoft PWB
Powersoft PowerBuilder
SCO
SunOS
SunSoft
Solaris
Symantec C++ IDDE
And many more...



Leverage Your Tool Investment with PVCS

There's a reason why all major Client/Server and LAN development vendors have formed alliances with INTERSOLV. It's PVCS—the standard for Software Configuration Management on the LAN. It's a vital ingredient for productive LAN development.

Every Major Development Tool Vendor Partners with Intersolv

The list on the left is only partial. PVCS is available in all the environments developers work in—today and tomorrow. That means less training—you don't have to learn another SCM tool if there's a change. That means more efficient development and easily enforced standards. It means consistent audit trails for all development, and the ability to correct errors and anomalies at any time.

Serious Application Development Demands PVCS

The trend is clear. Important applications in your company are moving to the LAN. They need the control and reliability that PVCS provides. Your development tool vendors know that—that's why they recommend PVCS.



**Call Today for
More Information:
800-547-PVCS Ext. 500**



The Standard for Software Configuration Management on the LAN

Inside U.S.A. • 1700 NW 167th Place • Beaverton, OR 97006 • 503-645-1150 • FAX 503-645-4576 • E-Mail PVCSINFO@INTERSOLV.COM
Outside U.S.A. • Abbey View • Everard Close • St. Albans • Herts AL1 2PS • United Kingdom • Tel 0727 812 812 • FAX +44 727 869 804

Circle Reader Service Number 2

We didn't say COBOL Workbench is the best in the world. You did.



PC SOFTWARE
BRAND
PREFERENCE

DOS-Based
CASE

AMONG TOTAL RESPONDENTS
Question: Which brand of DOS-based
CASE software are currently installed in
your organization?

INSTALLED IN COMPANY	EASY TO USE	BEST TECHNOLOGY	BEST PRICE/ PERFORMANCE
KnowledgeWare, Inc. (Information Engineering Workbench) 42%	Micro Focus (Workbench) 38%	Micro Focus (Workbench) 29%	Micro Focus (Workbench) 10%
Micro Focus (Workbench) 40%	KnowledgeWare, Inc. (Information Engineering Workbench) 17%	Texas Instruments (Information Engineering Facility (IEF)) 26%	KnowledgeWare, Inc. (Information Engineering Workbench) 14%
Intersolv (Facelator) 15%	Intersolv (Facelator) 9%	KnowledgeWare, Inc. (Information Engineering Workbench) 24%	Intersolv (Facelator) 11%
Texas Instruments (Information Engineering Facility (IEF)) 13%	Computer Associates (CA-Realia) 8%	Computer Associates (CA-Realia) 5%	Texas Instruments (Information Engineering Facility (IEF)) 11%
Computer Associates (CA-Telon) 7%	Visible Systems Corp. (Visible Analyst Workbench) 8%	Cadre Technologies, Inc. (Teamwork) 3%	Computer Associates (CA-Realia) 5%
Computer Associates (CA-Realia) 6%	Texas Instruments (Information Engineering Facility (IEF)) 4%	Intersolv (Facelator) 3%	Visible Systems Corp. (Visible Analyst Workbench) 5%
Intersolv (VSDesigner) 3%	Cadre Technologies, Inc. (Teamwork) 2%	Computer Associates (CA-Telon) 2%	Cadre Technologies, Inc. (Teamwork) 3%
Sybase		Sch...	

Question: For each of the DOS-based CASE software listed, please indicate which
company you most closely associate with each characteristic:

• Easy to use • Best technology • Best price/performance
• Best service/support • Best documentation • Prefer to do business with

Question: Which brand of DOS-based
CASE software are likely to be purchased
during the next 12 months?

BEST SERVICE/ SUPPORT	BEST DOCUMENTATION	PREFER TO DO BUSINESS WITH	PLAN TO BUY
KnowledgeWare, Inc. (Information Engineering Workbench) 26%	Micro Focus (Workbench) 14%	Micro Focus (Workbench) 28%	Micro Focus (Workbench) 50%
Micro Focus (Workbench) 26%	KnowledgeWare, Inc. (Information Engineering Workbench) 11%	KnowledgeWare, Inc. (Information Engineering Workbench) 23%	KnowledgeWare, Inc. (Information Engineering Workbench) 29%
Texas Instruments (Information Engineering Facility (IEF)) 19%	Intersolv (Facelator) 10%	Texas Instruments (Information Engineering Facility (IEF)) 19%	Texas Instruments (Information Engineering Facility (IEF)) 16%
Intersolv (Facelator) 10%	Texas Instruments (Information Engineering Facility (IEF)) 7%	Intersolv (Facelator) 6%	Intersolv (Facelator) 5%
Cadre Technologies, Inc. (Teamwork) 3%	Computer Associates (CA-Realia) 3%	Computer Associates (CA-Realia) 4%	Cadre Technologies, Inc. (Teamwork) 3%
Computer Associates (CA-Realia) 1%	Sybase	Computer Associates	Computer Associates



We've always believed Micro Focus COBOL Workbench® is the best in the world. Now you've confirmed it by unequivocally placing it first in the 1993 Computerworld PC Software Brand Preference survey.

You voted COBOL Workbench as **Best Technology** and **Easy to Use**. That's not all. Workbench has also won top honors in **Price/Performance**, **Best Documentation**, **Plan to Buy**, and most importantly, you chose Micro Focus as the company you **Prefer to do Business With**. According to you there was no contest.

Not surprising really. There is no better technology for developing new systems or re-engineering existing applications

on the workstation. Programmers find Workbench puts them directly in control of their development environment, delivering quality business applications on time and on budget.

If it isn't Micro Focus COBOL Workbench, it isn't in the running. That's not just our opinion, it's yours.

For your free copy of the 1993 Computerworld survey, or for more information, call 800-MF-COBOL, (800-872-6265).

Circle Reader Service Number 3

MICRO FOCUS

Micro Focus Inc. 2465 East Bayshore Road, Palo Alto, CA 94303. Tel. (415) 856 4161.

Micro Focus and COBOL Workbench are registered trademarks of Micro Focus, Inc. All other trademarks are property of their respective companies.

GSA Contract Number GS00K93AGS6403. In Canada call IBM Canada at 1-800-465-1234.



EDITOR'S COMMENTS

The Pen is Mightier Than the Mouse **5**



GUI CORNER

The Forbidden Text **6**



CORPORATE STUDY

Utility Company Adds Power with OS/2 **12**



PROGRAMMING INSIDER

More InSOMnia and Workplace Shell Programming **20**



OBJECT-ORIENTED PROGRAMMING

Component Software for the Masses: OpenDoc is Here **36**
An Example OS/2 C++ Class Library System **52**



PEN SYSTEMS

The Talking Pen: Coding a Pen/Audio Application **42**
A Natural User Interface: Writing a Pen-Centric Application **63**
OS/2 Survives the Pits **84**



COMPILERS

Compilers and Interpreters Buyer's Guide **74**



PRODUCT WATCH

New Tools for OS/2 **82**



RESOURCES

OS/2 Training Buyer's Guide **95**



Programmer's Paradise®

Call for a
FREE
Catalog!



SourceLink v2.0

by One Up Corporation

SourceLink, the ultimate 32-bit programming development tool, combines the functionality of hyperlink source code access, a fully-functional editor and extensive file manipulation utilities into one very powerful toolset. Now you can find code, change code, spawn compiles and hyperlink directly to the source of error quickly and efficiently. Features point & click source code navigation and automatic generation of function call trees.

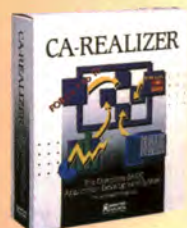


List: \$299 Ours: \$215 FAXcetera #: 6005-0003

CA-REALIZER

by Computer Associates

Defines a new generation of development tools that handles the mechanics of event-driven programming, message passing, process sharing and other complexities behind the scenes. Combines a structured superset of BASIC extended to access Windows and OS/2 objects and resources. CA-Realizer will help you create spreadsheets, charts, text editors, animation, graphics tablets and user-friendly forms from tools that can be created and manipulated by simple commands.



List: \$247 Ours: \$79* FAXcetera #: 1004-0008

* While supplies last.



Window Washer v2.0

by One Up Corporation

The latest version of the best-selling 32-bit screen saver for OS/2, with full system password security and the most complete monitor burn-in protection available today. Version 2.0 features many exciting animated effects, digital video and audio (plays CD's, MIDI or WAV files with program effects). Also utilizes TIF, GIF, BMP, & PCX backgrounds.

List: \$40 Ours: \$29 FAXcetera #: 6005-0001



SMART—Source Migration Analysis Reporting Toolset

by One Up Corporation

Hailed by IBM and other industry leaders as a premier source code conversion tool, SMART produces native 32-bit source code for OS/2. SMART runs on OS/2 2.1 or above and is used for the migration of 'C' and C++ source code from Windows 16-bit to OS/2 32-bit or OS/2 16-bit to OS/2 32-bit, depending upon the licensed, installed tables. The result is a faster, more tightly integrated application that does not require any emulation or run-time modules.

List: \$14,995 Ours: CALL FAXcetera #: 6005-0004

object-Menu

by Lifeboat Publishing

object-Menu is the way to quickly create powerful object-oriented applications. Built-in aesthetics make it easy to create interface styling such as Windows, Motif, or your own custom design. Portability to DOS, Windows/NT and OS/2 enables you to offer your product to multiple target markets with a single engineering effort. And, object-Menu's intuitive architecture, straightforward methodology and Visual Design tool significantly speed GUI development to allow you more time to focus on your application.



List: \$299 Ours: \$269 FAXcetera #: 2088-0003



WATCOM VX•REXX & WATCOM SQL for OS/2 by WATCOM

WATCOM™ VX•REXX is an award-winning easy to use visual development environment for creating applications that leverage the capabilities of OS/2 2.x and exploit the Presentation Manager graphical user interface. WATCOM™ SQL for OS/2 is a high performance SQL Client/Server DBMS for OS/2 which includes a standalone single-user SQL database server as well as a variety of interfaces to access WATCOM SQL from many popular OS/2, Windows and DOS applications. WATCOM SQL for OS/2 also includes a high-level interface for REXX, allowing easy access from WATCOM VX•REXX applications. Using the VX•REXX visual designer, you can graphically create Presentation Manager interface objects, quickly customize their properties, and easily attach REXX procedures to the objects using powerful drag-and-drop programming techniques. These procedures can easily retrieve information from the database for use in the application. Key features of WATCOM SQL for OS/2 include: referential integrity; bi-directional, scrollable, updatable cursors; row-level locking; ANSI SQL and IBM SAA compatible; multiple simultaneous application connections; symmetric multi-threading of concurrent requests; and the ability to import data from popular file formats including DBF.



SPECIAL BUNDLE OFFER!

Only: \$349 FAXcetera #: 1683-0019

GUARANTEED BEST PRICES! (Call for details)

To order call: 800-445-7899

Corporate: 800 441-1511

FAX: 908 389-9227

International: 908 389-9228

Customer Service: 389-9229

Programmer's Paradise Italia:

39-2-96702855

For more information on the products featured on this page call—

FAXcetera®: (908) 389-8173

Programmer's Paradise®

1163 Shrewsbury Avenue
Shrewsbury, NJ 07702

• All prices are subject to change without notice.
• Call for details on return policy and shipping charges.

Circle Reader Service Number 4

9
8
7
6
5
4
3
2
1
0
0
0
8

EDITOR*Dick Conklin***EXECUTIVE EDITOR***Nicole Freeman***MANAGING EDITOR***Marta Dils***EDITORIAL ASSISTANT***Diane Anderson***EDITORIAL ARTIST***Peter Altenberg***GROUP DIRECTOR***Regina Starr Ridley***PUBLISHER***Cathy Passage***ADVERTISING SALES***Angela Barnett**(415) 905-4983**Holly Meintzer**(212) 626-2275**Kristin Morgan**(212) 626-2498***MARKETING MANAGER***Susan McDonald***ART DIRECTOR/MARKETING***Christopher H. Clarke***ADVERTISING PRODUCTION****COORDINATOR***Tom Marzella***VICE PRESIDENT, PRODUCTION***Andy Mickus***VICE PRESIDENT, CIRCULATION***Jerry Okabe***CIRCULATION DIRECTOR***John Rockwell***CIRCULATION MANAGER***Lucinda Formyduval***SUBSCRIPTION INQUIRIES***U.S.: (800) WANT-OS2**Foreign: (708) 647-5960***BY DICK CONKLIN**

Editor's Comments

The Pen is Mightier Than The Mouse

In this issue, we will look at a growing set of OS/2 programs that require neither a keyboard nor a mouse: pen-aware applications. You have probably noticed the growing number of pen-input systems in use today—when was the last time you were asked to sign your name on a computer display? Did you know that OS/2 is pen-aware (thanks to IBM's Pen for OS/2) and that your applications can easily be adapted for this new market? Here's your chance to find out how, thanks to contributors Sarka Martinez, Kevin Lee, and Vera DuLaney.

Pen-aware is fine, but pen-exploitative applications really open up some possibilities. Nothing is user friendly about the trading pits on the floor of the Chicago Mercantile Exchange. Ordinary hand-helds and ordinary operating systems just won't survive the high-pressure, real-time demands of futures trading. But OS/2 does just fine. Larry Gavin and his Chicago CUBS provide an insider's view of a dynamic application.

OBJECT PROGRAMMING IS HERE

Our September/October issue will feature object-oriented programming. We couldn't wait until then to get started, however, so we asked Robert

Tycast, a long-time OS/2 Developer contributor, to tell us about OpenDoc. This evolving standard is what object technology is all about. It won't be long before we are all working with compound documents—especially those of us in the publishing business! IBM, Apple, and WordPerfect have already seen the future and are hard at work making OpenDoc a reality.

*Dick Conklin*

Contact	CompuServe	Internet
Dick Conklin (Editorial)	76711,1005, or OS2DF2 Forum	os2mag @vnet.ibm.com
Miller Freeman (Circulation)	71572,341	71572.341 @compuserve.com
Cathy Passage (Publisher)	71673,1163	csp @mti.com
Marta Dils (Managing Editor)		mdils @mfi.com

IBM OS/2 Developer: Vol. 6, No. 4, July / August 1994

Subscription information: U.S.: One year (six issues), \$39.95. Canada, Mexico, and international surface mail, add \$16 per year for postage. Canadian GST no. 124513185. International air mail, add \$30 per year for postage. Foreign orders must be accompanied by payments in U.S. funds. For new orders and customer service, call (800) WANT-OS2 (800-926-8672) or (708) 647-5960 (fax: 708-647-0537). IBM employees and branch office customers can subscribe to OS/2 Developer through IBM Mechanicsburg's Systems Library Services (SLSS) using OS/2 Developer's order number: G362-0001. POSTMASTER: Please send address changes to OS/2 Developer, P.O. Box 1079, Skokie, IL 60076-8079. Allow eight weeks for subscriptions to begin. OS/2 Developer (ISSN 1073-0729) is published bimonthly by Miller Freeman Inc., 600 Harrison Street, San Francisco, CA 94107, (415) 905-2200. Application to mail at second-class postage rates is pending at San Francisco, CA and additional mailing offices.

OS/2 Developer has applied for BPA membership.

© Copyright 1994 by Miller Freeman Inc. PRINTED IN THE U.S.A.

Miller Freeman, Inc.
A MEMBER OF THE UNITED NEWSPAPERS GROUP



This is the third of a series showing how a replacement list box smashes the 64K barrier and illustrates new methods of control design. In this issue, we cover the basics of text editing. **By MARK A. BENGE and MATT SMITH**

The Forbidden Text



Mark A. Bengé



Matt Smith

In previous columns, we began the presentation of a 32-bit replacement to the list box control. We mentioned that we would add direct editing to the list box. Before we attempt to do that, we will take a detour into the basics of text editing. To illustrate this side trip, we develop a replacement to the entry field. If you understand the basics of editing text, it will be much easier to add this support to the list box.

WHAT'S EVERYBODY AFRAID OF?

It is extremely difficult to find any books or articles that discuss the issue of text editing within a graphical environment. Just because the default text is proportional doesn't mean we should run away and hide. It's easy to create a text input mechanism for fonts of fixed increments, like Courier, but what are you supposed to do when it is a proportional font? You would think that by this time someone would have documented the simple—yes, simple—tricks that allow you to do it.

BACK TO THE FONT BASICS

Why hasn't the issue of text editing been adequately documented? Basically, it's because each of the letters has a different width. Figure 1 shows the fonts used within an OS/2 command window. The

font widths are all the same, and each font increment is a multiple of that width.

Look at Figure 2, which shows the system font. Some font widths are the same, while others are larger or smaller—hence, the big scary monster. In each figure, the running position is shown across the top of the font, and the width of each character is shown at the bottom.

WHERE DID THAT MOUSE GO?

Let's assume we use the traditional method to determine the character index based on the location on which the mouse pointer was clicked. If the pointer was clicked in position 112, we would use the following formula to determine which character is selected:

$$\text{xClick} / \text{CharWidth} = \text{CharIndex}$$

Therefore, $112 / 12 = 9$ or the character *j*.

What happens if we use this formula with a proportionally spaced font? If the average character width is 8, the formula yields $112 / 8 = 14$ or the character *o*, which is wrong. It should be the character *m*.

TRANSLATION, PLEASE

What's the trick? The answer is one of two methods. The first method, which is the fastest but contains a memory overhead,



is to build an array of points based on the text string. Conveniently, there is a function within the GPI that will do this for us.

`GpiQueryCharStringPos` is used to query the starting point for each character within a string passed to the function. The points are returned in an array of `POINTL` structures that is passed to the API. We use this array to determine the character position.

In the previous example, if we passed to the function the exact character string shown, the points would be returned in the `POINTL` array, and we would only have to look at each array element and the next element to determine if the point is between each. If it is, we have found the character.

The alternate method is to use the `GpiQueryTextBox` or the `WinDrawText` APIs, while at the same time shifting the number of characters you view. The alternative, while a memory miser, can be much slower, depending on the method you implement to perform the calculations. Remember, every time you invoke a system API, you don't know how many lines of code that API will execute.

The fastest method of performing

the translation from the point to the character index is to perform a simple binary search until the character cell is located. Both translation methods could employ the technique.

In our replacement for the entry field, we employ the first method of using the `GpiQueryCharStringPos` and the binary search technique.

A MATTER OF SELECTION

How do we translate this into reality with the replacement entry field? Quite easily, actually. Figuring out the character index is a piece of cake; handling the selection mechanism is a different story.

Figure 3 shows the basic design of the entry field. In terms of character selection, we are concerned only with horizontal locations. This is due to the nature of the control—that being a single line entry field. A simple click of the left mouse button causes us to look up the character index based on that location. One subtlety is how each click is translated to the final character index.

When the mouse pointer is clicked, the look-up routine determines the character over which the click occurred. To



Figure 1. Fixed font metrics



Figure 2. Proportional font metrics

Tales from the Trenches

A Public Service Announcement: When we started developing the replacement entry field, we decided to throw in cursor management calls. Bad move...or so it seemed. Although we specified in the `WinCreateCursor` call that the cursor should flash, it would not flash. For three solid days, we smacked our heads against the display, desperately trying to figure out what we were doing wrong. We finally figured it out.

We set a breakpoint on the `WM_TIMER` message to see if it was intercepting a timer message for the flashing of the cursor. Sure enough, the breakpoint was hit. After looking at the timer ID, we knew it wasn't one of ours that was set off by mistake. Because of the sheer diligence with which we have scoured the Presentation Manager headers over the past few years, we understood that this idea made sense. Buried within the `PMWIN.H` header are the IDs of four timers used within Presentation Manager.

One of them is `TID_CURSOR`, which was the ID of the timer we encountered at the breakpoint. The solution to getting the cursor to flash again? We put in a couple of lines of code that check the ID of the timer, determine whether it is the cursor ID, and pass the message on through to the `WinDefaultWindowProc`. Guess what? This isn't documented anywhere in the Presentation Manager documentation. If you follow the `WM_TIMER` reference, you are supposed to return zero (0). Maybe the Presentation Manager ID groups are spending too much time at the beach in Boca.

facilitate drag selection, each character is divided into two zones. The front half of the character is used to select the character. The back half of the character is used to select the next character. Hence, you have a conceptual click zone like that shown in Figure 3.

Why is this done? The answer is based simply on boundary conditions. This mechanism makes for a smoother selection when clicking on a character. It also makes the dragging selection mechanism appear smoother. Who said appearances aren't everything?

ANCHORS AWAY

The selection of text is based on a zone with an anchor point. The zone starting and ending points

make sense. What is the anchor all about?

The drag selection of the text can be performed in either direction: left to right or right to left. The anchor is the point on which the selection zone flips when the drag selection changes from one direction and reverses to the other.

This allows the starting and ending selection points to be non-directional in nature. The anchor is always based on the initial mouse click position. Therefore, when the mouse is dragged from left to right, the anchor is, in effect, the starting position of the selection zone. If the mouse is suddenly dragged from right to left and in front of the old starting location, the anchor is now, in effect, the ending position.

When you use this technique, you ensure a smooth transition from one zone to the other.

Depending on the direction of the dragging, the cursor is located either at the beginning or the end of the selection zone. When dragging from left to right, the cursor always ends up at the end of the zone (this is where the mouse release occurred). The cursor is placed at the beginning of the zone when the dragging is right to left.

CHARACTER REFERENCES, PLEASE

You can perform different operations on this selection zone. The most common action is deleting selected text and replacing it with a text string (a paste operation) or a single character (a keyboard character keystroke). You also could cut or copy the selected text to the clipboard if you desire.

By default, the entry field begins in insert mode. When no selection zone is active, the cursor location becomes the focus of interest. Any text pasted into the entry field causes the text to the right of the cursor to be pushed to the right. A simple character keystroke from the keyboard achieves the same result.

When the entry field is in over-type mode, the character positioned immediately after the cursor is selected. If a character keystroke is performed, the selected character is replaced with the new one.

SCROLLING THROUGH THE PARK

Once we have the basics in place, the fun begins. Usually, when you perform drag selection, you need to force the text that is not displayed to be shown.

The most often used technique is to move the mouse pointer to the left or the right of the entry field, while the left mouse button is continuously depressed. Once outside the left or right limits, the

PRESTO! A PRACTICAL CLIENT/SERVER SOLUTION!

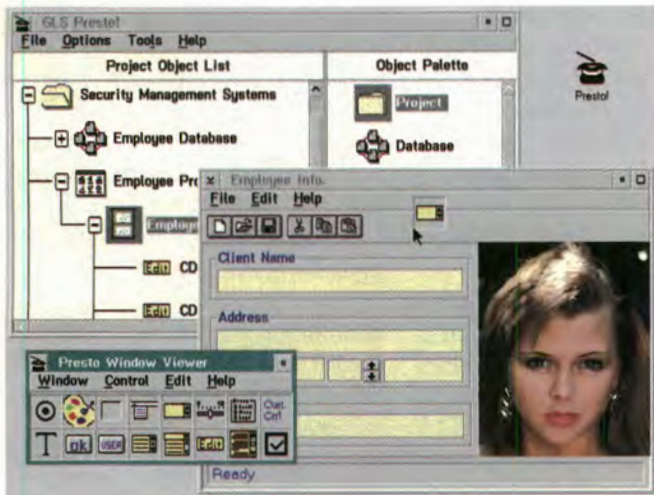
Introducing GLS-Presto 2.0.

Now you can build state of the art OS/2 and Windows applications for hundreds and even thousands less.

Presto is the only client-server development tool which generates OS/2 and Windows applications in COBOL, C and C++, for one low price.

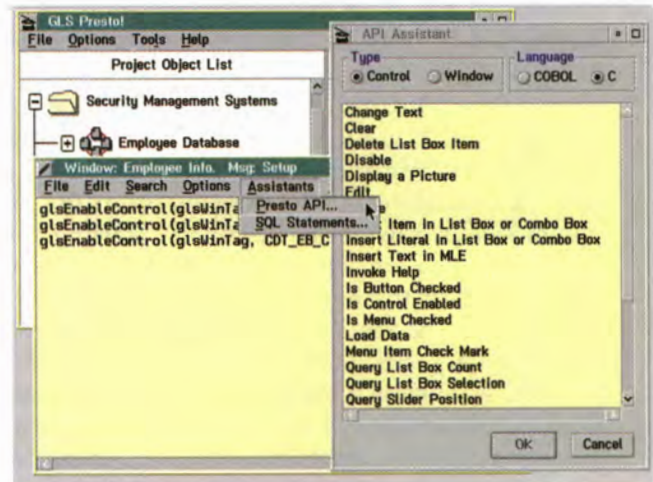
Presto is designed to help you become a client/server magician as quickly as possible without compromising on power, flexibility and security.

With Presto there is no need to invest in proprietary technology or suffer through an extensive learning curve. Presto allows you to take advantage of COBOL, C, C++ and SQL while effectively isolating you from the complexities of client/server development.



Presto's intuitive, easy to use integrated development environment will have you performing client/server tricks in no time at all. Design, prototype, generate, test and debug without ever leaving the IDE.

Presto makes the difference between OS/2 and Windows disappear, and lets you take advantage of the best of what they have to offer. For example, Windows MDI support is included for OS/2 and OS/2 control types may be used within Windows. Simply define your application, then select the target environment when you are ready to generate. Presto does the rest.



- Create complex applications in a minimum of time and run them under OS/2 and Windows.
- No knowledge of GUI or SQL programming required. The Presto code assistants are always ready to serve.
- Supports RAD methodology with instant proto-typing.
- Generated applications are quick and compact.
- Works with all major COBOL, C and C++ compilers.
- Incredibly fast. Applications generate in seconds.
- No compromises. MDI, tool bars, status bars, window geometry, custom controls and more.
- Use any DBMS or file access method compatible with your compiler.
- Fully networkable. No cumbersome object management required.
- Multi-media capability built in. Display GIF, TIF and Bitmap images dynamically.
- Absolutely no royalties or run-time fees.

Presto can appear on your computer for only



There is no better value.

To make some magic of your own, give us a call now at (219) 481-5809. Or, place an order via CompuServe, our mailbox is 73762,114.

idea is to allow the scrolling of the contents of the entry field in that direction.

What happens when the mouse isn't within the entry field window? In our previous article, "Scrolling Through the Park" (Winter 1993), we discussed the technique of capturing the mouse using `WinSetCapture`, such that when the mouse is outside the confines of our window we still receive the movements of the mouse pointer. But what if the mouse isn't moving? Tick, tock, tick, tock...

Yes, we rely on a timer that we start for the scrolling. Here, we are performing more magic tricks.

When we start the scrolling through the timer, we use a slower time period. Once the first timer message for the scrolling is received, we halve the time so that the scrolling seems to immediately speed up. This effectively gives the drag selection a smoother look.

SHOWING OFF

It is difficult to display the selections within the entry field. Here, different conditions apply. Referring again to Figure 3, notice that the text is divided essentially into three display zones. The first part is displayed normally. The second part is displayed with highlighting. And the last part is shown normally.

Unfortunately, there is no API within OS/2 Presentation Manager that allows us to display text in this

fashion (but we wish there was!). You must design the display routines so they subdivide the text into the three component parts, such that the drawing appears sequentially. If you don't do this, you may get a rougher drawing of the text in which portions appear to flash.

When you take into consideration the effects of offsetting the starting point of the scrolled text, things do become very interesting.

Basically, the drawing components of the entry field are divided into different drawing routines. Each routine is designed for speed instead of a general purpose engine.

Since it is so frequently used, this area can be continuously reworked so that every ounce of power is squeezed out.

WHAT, YOU DON'T LIKE THE MULTILINE EDIT CONTROL?

With this entry field replacement, you now have the basis to implement many different editing components. In our column that will appear in *OS/2 Developer's* November/December 1994 issue, we will take the principles of the entry field and apply them to the list box.

How could we further enhance this control? What about replacing the multiline edit control? Rather than replace it by providing exactly the same functionality, we could provide an enhanced entry field that can handle multiple lines.

This latter approach seems to make more sense. Here, you would just rely on the idea of `\n` separat-

ing each line within the control. The scroll bars could be provided to allow scrolling similar to that found in the multiline edit control.

By providing a few messages and allowing the entry field to have additional lines, you could easily create your own multiline edit control without much grief. It's easy now that you know how to translate that mouse click into a character selection.

In a future column, we will provide answers to your Presentation Manager programming questions. If you have a problem you would like us to address, please contact us at our e-mail addresses.

Mark A. Benge, IBM Software Solutions, Cary, N.C., is a staff programmer who joined IBM in 1989 and has worked on various CUA '91 controls for OS/2 2.x. He works in IBM user interface class library development, where he was involved in the implementation of C++ classes for drag and drop in C Set++ 2.1. Benge has a B.S. in computer science from Western Carolina University. He can be reached on CompuServe at 73532,2063 and on Internet at banzai@vnet.ibm.com.

Matt Smith, Prominare Inc., Toronto, Ont., Canada, is lead architect for the Prominare Development System, an OS/2 2.x advanced GUI development environment. He has been actively involved with OS/2 since 1988 and cofounded Prominare in 1990. Smith has a degree in architecture from the University of Waterloo. He can be reached on Internet at prominar.io.org.

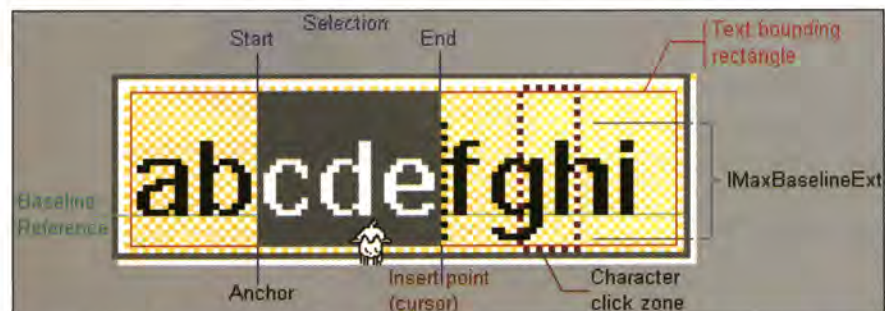


Figure 3. Entry field design

REFERENCES

You can download `EDCTL1.EXE` from these electronic sources:

CompuServe's OS2DF2 forum (OS/2 Developer section).

File area 11 on the IBM PCC BBS, (919) 517-0001.

OS/2 tools section on OS/2 BBS for IBM TALKLink customers.

Now, all the C/C++ development tools you need come in one handy package.



Bonus Bundle: C Set ++ CD-ROM, OS/2 for Windows, KASE:Set and Experience C++...just \$249!



If you want it all, you've come to the right place. IBM C Set ++™ Version 2.1 is the most comprehensive C/C++ development package you can buy for OS/2®. Period.

C Set ++ can dramatically slash your programming time with an intuitive graphical interface and world-class optimization. Your applications are quick to build, fast to execute and easier than ever to modify.

C Set ++ brings you the most complete set of class libraries and tools available for OS/2 development. It includes:

- A User interface class library and Collection Classes
- Visual debugger and Execution Trace Analyzer
- Class Browser and WorkFrame/2™ Version 2.1
- The IBM Developer's Toolkit for OS/2

...everything you need to make C/C++ development faster, easier and more hassle-free than ever before. And where C Set ++ stops, our 24 hour comprehensive IBM service and support begins. We're on call, when you call.

Now, when you purchase the C Set ++ CD-ROM, you will also receive OS/2 for Windows™, the KASE:Set™ GUI-builder and the revolutionary Experience C++ tutorial... all for just \$249.

KASEWORKS™ KASE:Set provides an introductory level GUI-builder, speeding the pace of your developments... and your learning.

Experience C++™ is the only leading-edge multimedia tutorial that lets you look and listen to detailed descriptions of every aspect of C++. To order your C Set ++ Bonus Bundle call our toll-free order hotline today!



ORDER TODAY
1 800 342-6672

\$249

For the IBM C Set ++ for OS/2 CD-ROM Version 2.1.

CD-ROM with hardcopy documentation is available for \$279, 3.5" disks with hardcopy documentation are available for \$309.

© 1994 International Business Machines Corporation.

IBM and OS/2 are registered trademarks of International Business Machines Corporation.

C Set ++ and Experience C++ are trademarks of International Business Machines Corporation.

KASEWORKS and KASE:Set are trademarks of KASEWORKS Inc.

Windows is a trademark of Microsoft Corporation.



Corporate Study

A natural gas company discovers that moving to OS/2 doesn't mean abandoning years of COBOL experience. **By BRIAN PROFFIT**

Utility Company Adds Power with OS/2



Brian Proffit

How do you update ancient mainframe COBOL environments to provide better productivity and lower maintenance costs? This is a problem many companies now face. Questar Service Corp., which provides information technology services to all branches of the Questar Corp. holding company, knew it needed to make changes. But going from a 3270 environment to a client/server solution with no downtime and minimum risk is challenging.

CUSTOMER INFORMATION SYSTEMS

For utility companies, customer information systems are key. Questar's customer information system covers straightforward name and address information kept by other industries. It also covers other aspects of the natural gas industry necessary for servicing more than a half-million customers in the northwest from its Salt Lake City headquarters. Questar Service employs over 200 people to handle the data processing needs of this organization. Since 1966, its customer information system has grown to over one million lines of mainframe COBOL code, covering data access and manipulation functions for customers, natural gas distribution, marketing, accounting, and the maze of required government regulatory information.

Over time, of course, Questar patched and added to the system. Maintenance became a nightmare. Jerry Edwards, director of Data Processing, reports that maintenance alone was costing around \$2 million per year.

"Adding five characters to an address field required changing over 300 programs, took nearly a year, and cost a horrendous amount of money," Edwards says. Clearly, a change was needed.

NEW DEVELOPMENT PROCESS, NEW BUSINESS PROCESS

In analyzing the existing spaghetti code, Questar discovered that the developers alone couldn't be blamed. Like most companies, Questar's business processes had also been added to and patched over time, leading to needless complexity and duplication of effort. To develop a clean, new information system, it also made the commitment to reengineer its business.

This decision made changes even more risky. By moving users from one system to another, Questar faced many potential problems, such as user difficulties and bugs, during the break-in period as the new code was exercised. Changing the business process at the same time doesn't double the risk; it



squares it. To lower the risk, Questar chose to migrate to the new level one piece at a time.

"We looked at the 54 subsystems containing around 700 programs and broke the whole project down into chunks that could be written and tested in no more than 90 days," Edwards says. "That way, the changes to the users would come in small, manageable pieces, and we could reengineer the business at the same time."

For the business changes, Questar decided to learn from the best. "We looked at companies all across the country to see how they did things," Edwards reports. "At each phase, we implemented the best technique we had found among them—changing the way the business did the process before we introduced the new technology. We redo things to remove duplications, for example. That has cost-justified the project all by itself."

This migration technique helped dictate Questar's choice of PC platform. "We looked at Windows, OS/2, OSF, and X Windows," recalls Edwards. "We found that Windows just didn't have the horsepower and wasn't robust enough for a mission-critical application. Plus, the response time wasn't sufficient, nor was the crash protection. We didn't feel that OSF or X Windows was going to be mainstream enough."

"We decided on OS/2 for several reasons. A big [reason] was its ability to run DOS software so well. That allowed us to continue running our 3270 emulators well while users began using OS/2 programs."

They're also exploiting OS/2's ability to run programs written for DOS and Windows in other ways. For example, they've developed a new Executive Information System using Visual Basic.

"We have automatic links to Dow Jones, getting stock information every hour automatically and doing graphic comparisons," says Edwards. "We also get weather information and satellite maps because parts of our business are weather-sensitive. We integrate [the information] right into our system. We have a tie-in to automated news services, retrieving articles related to our industry. We can cut-and-paste those into our word processor or e-mail. The OS/2 architecture is giving us benefits we hadn't anticipated. Of course, just being able to do something else while a compile is running helps a lot."

OLD SKILLS, NEW TOOLS

Questar decided that, although it was revamping its software and business, changing everything didn't make sense. "We estimated it would have cost an additional half-million dollars in retraining costs if we were to change from COBOL to C. Then you have to factor in the huge amount of additional time, personnel problems, and risk of using code from programmers not familiar with the language."

By 1992, Questar had the tools in place that allowed it to take advantage of its programmers' comfort level with COBOL. One tool they used was the MicroFocus Workbench, an integrated development environment with several functions designed to help these migrations. For example, the Workbench can automatically convert EBCDIC to ASCII as mainframe code is brought to the workstation.

An important aspect of the change, however, was the greater productivity a consistent graphical user interface could bring the user. Questar wasn't interested in just bringing batch code to the workstation. But learning the Presentation Manager API and COBOL bindings sufficiently to produce high-



MAINFRAME AS FILE SERVER

The old, flat file structure was a large part of the maintenance difficulties because it didn't offer much flexibility. Questar felt it could salvage its knowledge of CICS but needed to move to DB2 databases.

"We wanted to set up a cooperative link between CICS/MVS and the workstation," Edwards says. "One of the reasons we selected OS/2 was the ease with which CICS/OS2 could be linked with the mainframe. We came up with a strategy to [use] intelligent workstations [and] the mainframe as a file server. The PCs are Compaq 486 workstations that access a seven gigabyte DB2 relational database on the host."

Currently, they have approxi-

mately 700 PCs on 26 NetWare LANs connected to the Amdahl mainframe via a WAN. Managing the development and distribution of software across an environment like this can be a headache. Questar uses IBM's Netview Distribution Manager to handle this and reports success.

MEASURING THE BENEFITS

"We have moved to small, independent development teams," says Edwards. "Having each piece small enough to complete in 60 to 90 days has helped them see faster results. That helps morale. Programs are coming in ahead of schedule and under budget. Development time has been reduced dramatically. What typically would have taken 600 to 700

developer-days is now down to 200. That lets us be more responsive as things change. For example, we have just decided to build systems for natural gas vehicles. We can have it completely implemented in three to six months.

"The [Presentation Manager] code is easier to maintain than the mainframe code. Plus, the change in development methodology cut down on maintenance. We're finding fewer fixes—partially because the user is helping with the design and partially because the quality of the tools allows us to build things with fewer bugs."

Edwards describes the four keys to Questar's success as new methodology, new tools, new team concept (with greater auton-



OS/2 Visual Programming with GpfRexx™

Using expertise gained from years of creating the best Visual Programming Tools for C programmers, Gpf Systems has created a visual programming tool for everyone. By combining the easy to use but powerful WYSIWYG Interface Builder of Gpf with the easy to use but powerful REXX language that comes with every OS/2 system, now you can create your own PM applications, even if you've never used Gpf or REXX before. Point, click, drag, and drop to build striking PM applications that take advantage of the advanced features of OS/2.

Visual Programming

Visual GUI programming for REXX; the most natural way to program a Graphic User Interface.

Simple but Powerful

Simple to use but powerful WYSIWYG Interface Builder which can create basic or advanced applications without detailed system level knowledge.

Getting the Most from OS/2

Take advantage of advanced OS/2 features: Multi-Tasking, SQL DataBase, Multi-Media, Multi-Media Controls, Communications (APPC, CPM-C, EHLLAPI)

Risk FREE



FEATURES

- Point and click programming
- Multi-thread programming
- Nested menus to any level
- Fully integrated true WYSIWYG interface builder
- PM programming without system knowledge
- Support for CUA 91 controls
- Supports Multi-Media at design and run time
- Supports user designed custom controls
- Control fonts and colors for windows and controls
- Easy inclusion of Bitmaps and graphics
- Context sensitive help for windows and controls
- Point and click access to DB2/2 SQL databases - and REXX supported Embedded SQL access to DB2/2 APIs
- Multi-thread source level debugging
- Royalty FREE - Configurable run-time

Gpf
SYSTEMS, INC.

Order GpfRexx today for Just \$247.50
60 days money back guarantee
For information on other development products call,
Phone: (800) 831-0017 or Fax: (203) 873-3302
30 Falls Road, Moodus, CT 06469



The Path To A Good GUI Isn't Always Clear

Navigate the maze with the NEW Gpf 2.1 Professional Developers Toolkit

Promises are cheap, GUI development tools aren't. Gpf 2.1 demo software is FREE. Try Gpf before you buy any tool and you'll agree that Gpf 2.1 DELIVERS.

With this powerful point and click visual programming environment you can create a CUA '91 Graphical User Interface for OS/2 Presentation Manager® or MS DOS/Windows® in as little as 10% of the time required to hand code the same design.

What You See Is What You Get programming reduces weeks of coding to hours. Quickly prototype and test your interface, then let Gpf write the code. Gpf generates error free, structured Object Oriented C or C++ source, complete with embedded SQL for OS/2 DataBase. Custom Help and Logic are added to controls as they are drawn to create full Client/Server or Stand-alone applications!

Gpf is hosted on OS/2 2.x and generates native code for OS/2 2.x, OS/2 1.3.x and MS Windows 3.0 or 3.1 Of course this means maximum performance with no run time module and no royalties!

* GUI Programming Facility

Seeing Gpf is believing!

- Design 32 or 16 bit GUIs for IBM OS/2® or Microsoft Windows®
- Be creative, WYSIWYG design environment
- Full CUA '91 Control set, 32 or 16 bit
- Full application and/or DLL generation
- Reusable Objects
- Automatic documentation
- Minimal time to application delivery



Gpf
SYSTEMS, INC.

Try us, FREE Demo Software Available

Order Gpf Pro Toolkit today for just \$1440.00

Separately: Gpf 2.1 for \$1295.00 & GpfTools for \$195.00

Or, try: Gpf 2.0 Single Platform, now available for just \$495.00

Call Gpf Systems Inc. at: (800) 831-0017

Phone (203) 873-3300 • fax (203) 873-3302 30 Falls Road, Moodus, CT 06469

Circle Reader Service Number 6

Circle Reader Service Number 7

omy), and a new management strategy. In fact, Questar was nominated for an Ultra Award—an honor bestowed on utility companies demonstrating significant technical innovation. Edwards was recently asked to discuss Questar's move to OS/2 at a customer information system conference sponsored by Price Waterhouse in Orlando, Fla.

"There was a lot of interest," Edwards says. "There are around 20 utilities [companies] that would like to come visit us and see what we're doing. A division of Questar is considering starting a consulting service."

MORE WORK, BUT FASTER RESULTS

The dramatic decrease in development time occurs even though Questar's technique demands more development effort. "We have made a commitment to keep the data in one place," Edwards says. "One of the advantages of OS/2 is that it allowed us to write some behind-the-scenes code to pass information between 3270 sessions and our OS/2 programs. Migrating small pieces has required us to build bridges between the pieces of data on the workstation and the pieces on the host. We still think that is better than having duplicate information. As we implement new pieces, we add new bridges and tear down some that are no longer needed. That is one of the hardest parts."

EASIER TRAINING AND SUPPORT, TOO

Questar has 18 offices running OS/2 workstations. Training and supporting users meant traveling to remote offices for tasks that required visual access to the user's computer. IBM's Distributed Console Access Facility/2 made a significant difference in the num-



Figure 3. Host-based data is tightly coupled with the workstation



Figure 4. Dramatically faster order processing linked with mobile terminals

ber of trips required and the duration of phone calls needed for troubleshooting.

"From 500 miles away, I can bring up a view of [a remote] system's display," Edwards says. "In fact, I can bring up more than one person's screen at the same time

in different windows. This lets [me] set up a conference call and monitor their system."

EXTENDED CONNECTIVITY

Questar has extended connectivity beyond its WAN. The company has a number of field per-

sonnel handling distribution of natural gas. The information in Questar's customer information system is accessible from the field via 135 mobile data terminals.

In fact, the mobile data terminals are processing over 8,000 transactions per day and have allowed Questar to reduce its dispatching personnel, saving about \$100,000 per year.

The mobile data terminals are a key reason why Questar cannot afford any downtime. They access the customer information systems 24 hours a day. This enhanced connectivity is also enabling better workload forecasting and scheduling of service orders.

THE MOVE CONTINUES

Questar is finding several new possibilities that the OS/2 workstations offer. For example, the engineering department is using an Intergraph computer mapping system. Questar is now integrating that into the Customer Information System, allowing access to over 5,000 maps.

Questar is taking advantage of Win-OS/2 functions as well, using Microsoft PowerPoint and Excel and WordPerfect for Windows. Questar is also evaluating Pioneer Software's Q+E query tool as a way of providing easy access to its database. To Edwards, this flexibility is key.

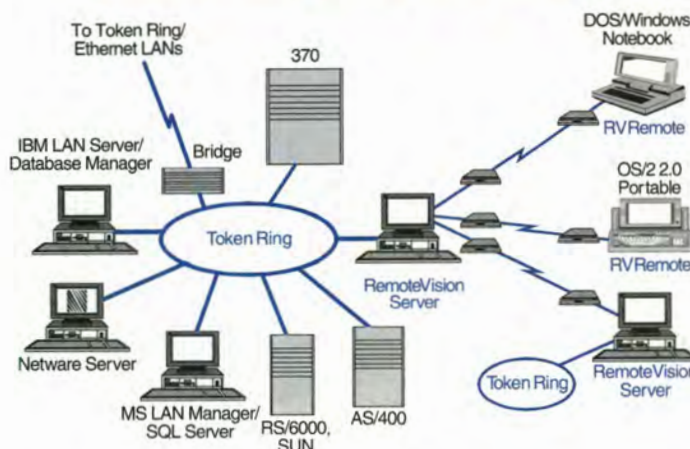
"We're strongly committed to OS/2, but that doesn't mean we can't run Windows applications on it. OS/2 is our lifesaver, allowing us to migrate and run the old and new at the same time."

Brian Proffit was part of IBM Corp.'s OS/2 development team before leaving to become director of PC Week Corporate Labs. Proffit is currently president of Visionary Research, an independent consulting firm. He also is the

author of two books, *OS/2 Application Development Tools* and *OS/2 Inside and Out*. Proffit is a contributing editor for *OS/2 Magazine* and has written for *PC Week*, *Dr. Dobbs' Journal*, and *Programmer's Paradise*.



RemoteVision™ MultiPort™ Servers. The Proven Solution For Remote Token Ring Networking.



- Extend LANs transparently over asynchronous modems to remote workstations and workgroups.
- Run DOS, Windows™, & OS/2™ apps written to multiple protocol stacks on remote nodes.
- Operate remote computers as full-function LAN nodes.
- Connect to remote offices with LAN-to-LAN dial-up networking.
- Add remote connectivity services using existing LAN machines.
- Access remote PC servers and hosts without additional host gateways.
- Up to 64 ports on a non-dedicated server

RemoteVision makes it easy and affordable for DOS, Windows and OS/2 users in remote locations to connect to Token Ring or other LANs using dial-up modems. Response time remains fast, applications don't need changing, and users won't need retraining. So you can turn your remote machines into full-function workstations. With RemoteVision. To immediately find out more, or to take advantage of our "Try Now, Pay Later" 30-day money-back guaranteed trial offer, call today.



1265 Montecito Ave., Ste. 101, Mountain View, CA 94043
Tel: (415) 965-8607, Fax info: (415) 965-8607, (then press 2)

CompuServe 71612, 1253. Trademarks are from their respective companies. ©1994 Token Technology, Inc. 001-1

Circle Reader Service Number 8

A Special Report from the publishers of OS/2 Developer!



Quantities are limited. Order now!

For fastest service, fax your order to us at 415-905-4967



☒ **YES! Please send me _____ copies of the REXX Report!**

Each copy is only \$9.95 in the US or \$10.95 in Canada.

Please add \$2.50 for shipping and handling.

☐ My check is enclosed

☐ Charge my credit card: ☐ Visa ☐ MasterCard ☐ American Express

Name _____

Signature _____

Date _____

Send them to:

Name _____

Address _____

City _____

State _____

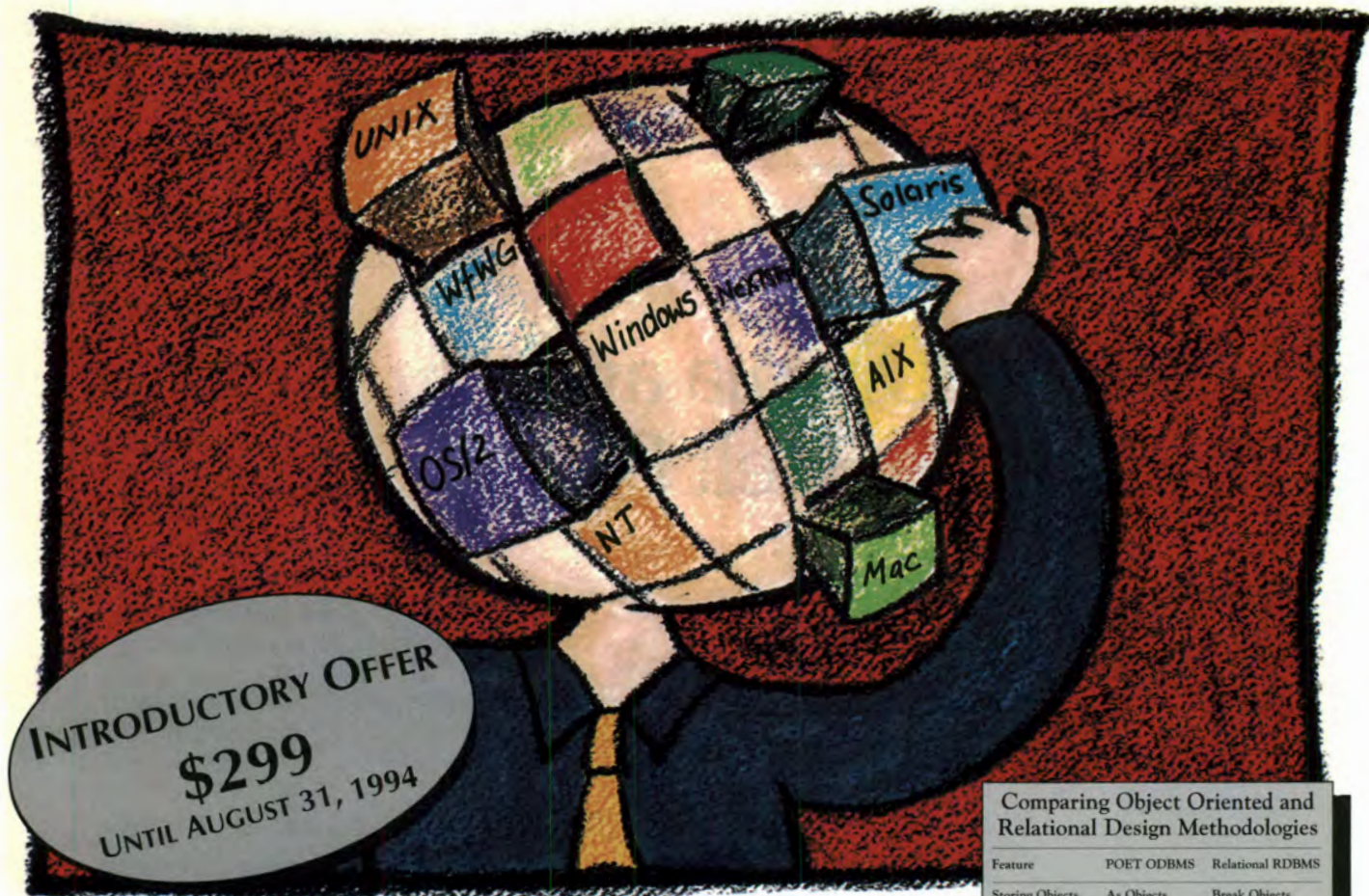
Postal Code _____

Mail: REXX Report
P. O. Box 7046
San Francisco
CA 94120

Phone: 1-800-444-4881

Fax: (415) 905-4967

Don't even think about using a Relational Database System !



Use the POET Object Database for C++

C++ and class libraries have made the GUI development much easier. After all, it is only logical that more and more developers think in objects. But object orientation shouldn't end at the user interface programming level.

The Problem: Without POET, a C++ programmer must use flat files or a RDBMS to store objects. He has to write code to overcome the mismatch between the application and the database model. This leads to design restrictions, performance penalties and more code to write and maintain.

The Solution: POET operates at the object level, it speeds up the development process and provides greater performance. Furthermore, the developer can simply take the object-oriented application design in C++ and map it 1 to 1 into the database. Without any compromise at all!

Full featured database:

POET provides complete support for Encapsulation, Inheritance, Polymorphism and Containers as well as queries, object locking, and transaction handling.

True cross platform support: Complete interoperability makes the development of network enabled applications a breeze. POET supports Windows 3.1, Windows NT, Win32s, Windows for Workgroups, Novell Netware (NLM), SUN, AIX, HP-UX, SGI, SCO, Macintosh, Power Macintosh, OS/2 and NeXTStep.

Comparing Object Oriented and Relational Design Methodologies

Feature	POET ODBMS	Relational RDBMS
Storing Objects	As Objects	Break Objects into Tables
Database Model	User Application Model	Separate Database Model Required
C++ Integration	Total	Poor
Database Operations	At Object Level	Must Write Code
Productivity	Increased	Reduced
Complex Object Performance	Excellent	Poor

Call 1-800-950-8845



POET Software Corporation, 4633 Old Ironsides Dr., Suite 110, Santa Clara, CA 95054, Tel.: (408) 970-4640 Fax: (408) 970-4630
 United Kingdom (Silicon River) +44 81-316-7777, Germany (POET Software) +49 40-60-99-00, Australia (Microway) +61-3-580-1333, France (LCI) +33-1-34-65-7777
 Netherlands (Protocols Software) +31 20 645 5023, Sweden (Duke Systems) +46 8 703 2781

Circle Reader Service Number 9



Programming Insider

Feeling a little Shell-shocked after last issue? This issue's Programming Insider finishes up in getting you on your way to writing effective Workplace Shell-exploitative code. **By DAVID REICH**

More InSOMnia and Workplace Shell Programming



David Reich

Sleeping better since last issue? Hopefully the Programming Insider column in *OS/2 Developer's* May/June 1994 issue helped you understand more about how OS/2's Workplace Shell fits in with OS/2 and running applications. In any event, this column takes that basic program a step further.

When I finish with these descriptions, you'll have an object that knows its type, its association types, and its icon. Also, the object will start the appropriate printing function when it is dropped on a printer object. You don't have to code all the function into the object, such as having it open, load, and print a data file. The object inherits some behaviors from the parent class, which I detailed last time.

In the last column, you saw what Workplace Shell objects can do and how you should use them, such as not writing an application as an object, but writing the object to represent the data to be manipulated by application programs. I also covered how the Workplace Shell fits into the OS/2 system in a large, complex, integrated application that was written using SOM.

You read about the basics of SOM and the SOM compilers and emitters, and how you derive everything from the .CSC file initially. From there, we took a trip into a small .CSC file that did

little but show how the files interrelate and provide an object to install and try out. I provided the `MAKE` file and a small piece of code that shows how to install an object.

THE CODE

Figure 1 contains the complete .CSC file for this object. For those of you who missed my last column, I will review the whole .CSC file and discuss the change from last time, which is the addition of the methods that are overridden.

After that, we'll take a look at the .C file shown in Figure 2. If you coded the example from the last installment and ran `MAKE` on it, you noticed that the .C file is generated for you by the SOM emitters. However, it looks empty, much like a window procedure that does nothing but call `WinDefWindowProc` for every message. In the .C file here, I added some of the filling. I will take you through the .C file function (or method) by function, describing what each does.

Then, you'll be ready to hack at your own objects and see what Workplace Shell can do for you.

THE .CSC FILE

The .CSC file is where it all begins. This file is the primary input to the SOM emitters. Depending on the SOM emit-



ter options you choose, many files are output. The most important output files are the C, H, and DEF files that, at the very least, are needed to build the DLL that will be the class.

Last time, the .CSC file contained only enough to define the class and show one example of a method to demonstrate what the .C file will look like. Now, the .CSC file in Figure 1 has all the method overrides to perform the basic level of work for any object.

At the beginning of the file is the `#include` statement for the parent class, in this case, `WPDataFile`. Following that is the class definition section with the class name, class hierarchy, and stems for the emitters (which will become apparent as we examine the .C file).

The formal parent declaration and the passthru section follow those sections. The passthru section is used to tell the emitters to pass the section straight through to the .H or .IH file (as specified in the passthru section declaration).

The other sections of the .CSC file define the instance data for the object (in this example, there is no need for instance data) and the methods. The methods section defines methods introduced by this object class, along with the methods the class overrides.

For this example, there are no methods that are introduced. You will notice, however, that there are two functions in the .C file that are not method overrides. Since the functions are not exported as methods, no other Workplace Shell object can invoke them. For this reason, they are private functions to the class.

The overrides are listed with instance method overrides first; class method overrides follow. Also note that there is a special `#define` in the .C file delineating the separation of instance and class methods. Since any new methods in the .CSC file will cause the .C file to be appended to when the SOM emit-

ters are rerun, you must move the declarations for new instance methods above this `#define`. Otherwise, your class will not function correctly.

THE .C FILE

The .C file appears in Figure 2. We'll look at the file function by function (or, more precisely, method call by method call). Before we get into the .C file, I'll recap what this object does.

The object is part of a mythical word processing package. The object class has a name and an icon. When a new instance of the class is created, it is associated with the executable program; when it is dragged over and dropped on a printer object, it will invoke the print code, passing it a flag to indicate not to show anything but to start up in "subset" mode to print the document. In addition, this word-processing document object class implements style sheets. There are several ways this can be done, and I include one in this example.

LET'S SEE WHAT THIS .C FILE IS AND DOES

First is the section for defines, includes, and various pragmas. Don't be afraid of these statements, although you may not ordinarily use them. The SOM emitters put most of them in there. The first few lines should look familiar. They are the define you set up in the .CSC file and some basic Presentation Manager includes. Then there is a function declaration that is internal to our class: `SetTypeEA`.

Next is a set of lines, added by the SOM emitters, that suspend the output of compiler warnings and include the .IH file. The reason for the suspension of warnings is that the prototypes in the .IH file can cause compiler warnings. Be aware that the suspension of these warnings can mask real warnings in your source, so you may want to run the

file through the compiler once without warnings suspended to see that you are not hiding anything that may be a problem later.

INSTANCE METHODS

First are the instance methods. These are the methods that the instances of the class respond to, as opposed to the class methods that the class responds to.

wpPrintObject

The `wpPrintObject` method is invoked on an object whenever it is dropped on a printer object. The default, or parent class's behavior for `wpPrintObject`, is to put up a dialog box that asks the user if the data file (the parent class of our class) is a plain text file or a printer-specific formatted file. If, as in our word processor's case, the file is neither, allowing the parent class's method to handle it will not enable drag-and-drop printing. We need a different approach.

If the data file format you use is neither plain ASCII nor printer-specific, and you want drag-and-drop printing, you will need to do some work. That work, however, is not very complicated. As outlined in the comments of this sample object .C file, you will need to call `wpQueryRealName` against the object to get the real file name the object represents. Then you will invoke the printing code of the application, passing that file name and some flag indicating that the application should print the file, but not show itself, and terminate upon completion.

One way to do this is to call `DosExecPgm` and invoke the application. Another way to accomplish this is dependent on the architecture of the application; but if you separate the printing code into a function or set of functions in a DLL, you can export the print interface from the DLL and call it from your object directly.

```

***** #
# Skeleton SOM/WPS object
#
#
# DISCLAIMER OF WARRANTIES. The following code is sample code. This
# sample code is not part of any standard product and is provided to you
# solely for the purpose of assisting you in the development of
# your applications. The code is provided "AS IS". ALL
# WARRANTIES ARE EXPRESSLY DISCLAIMED, INCLUDING THE IMPLIED
# WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
# PURPOSE.
#
*****

*****
# Include the class definition file for the parent class
# The parent class of this sample class is wpDataFile
*****

include <wpdataf.sc>

*****
# Define the new class
*****

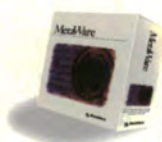
class: MyNewObject,
    external stem    = myn,
    local,
    external prefix  = myn_,
    classprefix      = mynO_,
    major version    = 1,
    minor version    = 2;

--
-- CLASS: MyNewObject
--
-- CLASS HIERARCHY:
--
--          SOMObject
--              WPObject
--                  WPFileSystem
--                      WPDataFile
--                          MyNewObject
--
-- DESCRIPTION:
--
--      Basic class definition that does nothing different from its parent.
--
*****
# Specify the parent class
*****

```

Figure 1. The new object's complete .CSC file (continued on page 24)

BREAK AWAY FROM THE COMPETITION



When you're pressing the limits of software performance, every microsecond counts. Efficient operation decreases execution time, putting you ahead of the pack.

That's why professional C/C++ developers use High C/C++. MetaWare's High C/C++ compilers offer eight levels of global optimization, plus specific optimizations for particular processors. The executables consistently out-perform competitors in benchmark testing.

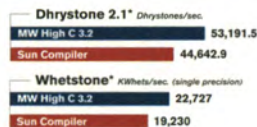
Besides offering rapid execution time, MetaWare's High C/C++ compilers provide a myriad of controls that allow you to customize and fine-tune code generation. There are command-line toggles and pragmas for customizing your application.

The compilers generate informative error and warning messages, and comply with major industry standards.

For years, MetaWare has supplied major OEMs

such as AMD, AT&T, CenterLine, IBM, Intel, and NCR, with 32-bit compilers for a variety of different processors and platforms. These OEMs depend on MetaWare for superior products and support.

Call (408) 429-6382, or email: techsales@metaware.com, for more information about features and specific platforms.



MetaWare

High C/C++ for OS/2, NT, DOS, Windows, AIX, UNIX SVR4, and Solaris

*Benchmark source and other performance information available upon request. All benchmarks use the -O option. MetaWare, the MetaWare logo, High C, and Professional Pascal, are registered trademarks, and High C/C++ and High C++ are trademarks, of MetaWare Incorporated. Other names are trademarks of their respective companies. Copyright 1994 MetaWare Incorporated, 2161 Delaware Avenue, Santa Cruz, CA 95060-5706.

Circle Reader Service Number 10

The method that uses `DosExecPgm` is the simpler of the two, both architecturally and programmatically. In it, you just call `DosExecPgm` and pass the information that is required. However, the DLL method is slightly more complicated and requires more work.

The reason I advocate the DLL method instead of the `DosExecPgm` method of printing is because the former is much faster and easier on system resources. You export your printing function from the DLL and code it so that you can pass a file name and flag to this exported function. Then, rather than starting up a whole new process with `DosExecPgm`, you will call the function in the DLL. Here, either you need to spawn a thread to call the print function or write the print function so that it starts its own thread inside the DLL to do the printing. If you do not do this, you will tie up the shell's thread doing your printing and, therefore, hold up processing in the shell.

Once the print job is spooled, you can terminate the program or function and return to the object, where you execute a return that ends the method override.

wpSetup

The `wpSetup` method is called when an object instance is created. You can view it as a `WM_CREATE` message in a window procedure. It lets you set up specific settings along with the object at creation time. In this example, I set the `.TYPE` extended attribute for the physical file.

To set the `.TYPE` extended attribute, you only need to define what the physical file name is and what the `.TYPE` should be. You do the former by calling `wpGetRealName` against the object itself. Once that is complete, you call the object class's class method for `QueryInstanceType` (`myn0_wpclsQueryInstanceType`), which, as you will see in a moment, is defined to return `My Object Document`.

```
parent: WPDataFile;

*****
# Passthru IMPLEMENTATION definitions to the .ih file
*****

*****
# Passthru PUBLIC definitions to the .h file
*****

passthru: C.h, after;

#define MyNewObject SOMAny

endpassthru; /* C.h */

*****
# Define instance data
*****

*****
# Define methods
*****

methods:

*****
# Define our own instance methods
*****

*****
# Define our own instance methods for setting instance data
*****

*****
# Define instance methods for querying instance data
*****

*****
# Specify instance methods being overridden
*****

/* Method being overridden */ /* Why we are overriding */
override wpPrintObject; /* Will call execpgm to print instance doc */
override wpSetup; /* To set up my icon */
override wpCreateFromTemplate; /* Will create dlg to ask which style file */
/* To use then use parent class to do the */
/* actual instance creation */
override wpCreateAnother; /* When another is created */
```

Figure 1. The new object's complete .CSC file (continued on page 25)



control that creation no matter what action the user takes to create a new document.

wpCreateAnother

The `wpCreateAnother` method is called when the user creates a new class instance by selecting "Create Another" from the context menu from another document object. When a user creates a new instance in this manner, you should have control over how the object is set up.

Whenever a new document is created in my word processor's world, I want to provide a dialog to the user, giving them a choice of style sheets. (You can do whatever initialization you like in here.) You can implement style sheets in any way you want. I recommend that you put up a file dialog, prompting the user for a file for use as a style sheet. In this way, you provide the flexibility of using any valid file as a style sheet; the user simply chooses it when creating a document through the Workplace Shell. You may also add a selection for no style sheet if you want. This is all up to you. The point is that you are performing some setup whenever the user creates a new instance.

To easily provide this style sheet selection, write a function that performs it. You will likely need to call it from multiple places, given the variety of ways you can create a new object instance.

`PutUpStyleSheetDialog`, a function that contains `DosBeeps` for demonstration purposes, is called inside the override to `wpCreateAnother`. However, this is done after a call to the parent method. This is so the parent method can actually create the instance. Then, our function takes over to put up the style sheet dialog. Once that is complete, the method is complete and the return is executed.

```
*****
#      Define class methods
*****

*****
#      Specify class methods being overridden
*****

/* Method being overridden */      /*      Why we are overriding */

override wpclsQueryTitle, class;      /* To give my class a title */
override wpclsQueryInstanceType, class; /* To set up a file object type */
override wpclsQueryInstanceFilter, class; /* To set up a filter for */
                                         /* instance data */
```

Figure 1. The new object's complete .CSC file (continued from page 24)

```
/*
 * OS/2 Work Place Shell Sample Program -
 *
 *
 * DISCLAIMER OF WARRANTIES. The following [enclosed] code is
 * sample code that is provided to you
 * solely for the purpose of assisting you in the development of
 * your applications. The code is provided "AS IS". ALL
 * WARRANTIES ARE EXPRESSLY DISCLAIMED, INCLUDING THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE.
 */

#define SOM_NoTest
#define MyNewObject SOMAny

#include <string.h>
#include <malloc.h>
#define INCL_PM
#define INCL_DOS
#include <os2.h>
```

Figure 2. Sample object's .C file (continued on page 26)

The extended attribute is needed to enable the associations. This is so the associations work properly when using the shell functions natively and from the `ASSOCTABLE` in the program executable file.

Now you have the file name and the value to be set in the

extended attribute. `SetTypeEA`, the internal function defined at the bottom of the program, will accomplish this.

The other two overridden instance methods, `wpCreateAnother` and `wpCreateFromTemplate`, are overridden. This is because you should


```

BOOL SetTypeEA( PSZ pszFilePath, PSZ pszEAValue );

/*
 * SOM generates method stubs that
 * cause compiler warnings. Suspend warnings
 * (but be wary, this can mask errors in the
 * emitted "passthru" sections of the CSC file).
 */

#define MyNewObject_Class_Source
#pragma checkout(suspend)
#include "mynewobj.ih"
#pragma checkout(resume)

/*
 *
 * OVERRIDE: wpPrintObject                ( ) PRIVATE
 *                                           (X) PUBLIC
 * DESCRIPTION:
 *
 *      Called to print a view of the object
 *
 *      Passed is a pointer to me, along with a structure that can be
 *      used to call DevPostDeviceModes or DevOpenDC call to be able to
 *      open a printer DC. When called here, I need to call DosExecPgm on
 *      my EXE, pass it the name of the file I represent and the print
 *      destination, along with a flag to tell it to now show itself, and just
 *      print the file on the destination. It is up to the app if it wants to
 *      also DevPostDeviceModes to see if the user wants to change any of the
 *      job properties from the defaults. (The app should go away after the
 *      doc is printed, but that is an app function. The "Best" way to do it
 *      is to have an entry point in one of your DLLs so you don't have to
 *      invoke the whole app. The entry point could spin off a print thread
 *      to be able to return control back to the shell so you don't freeze the
 *      shell while the doc prints.
 */

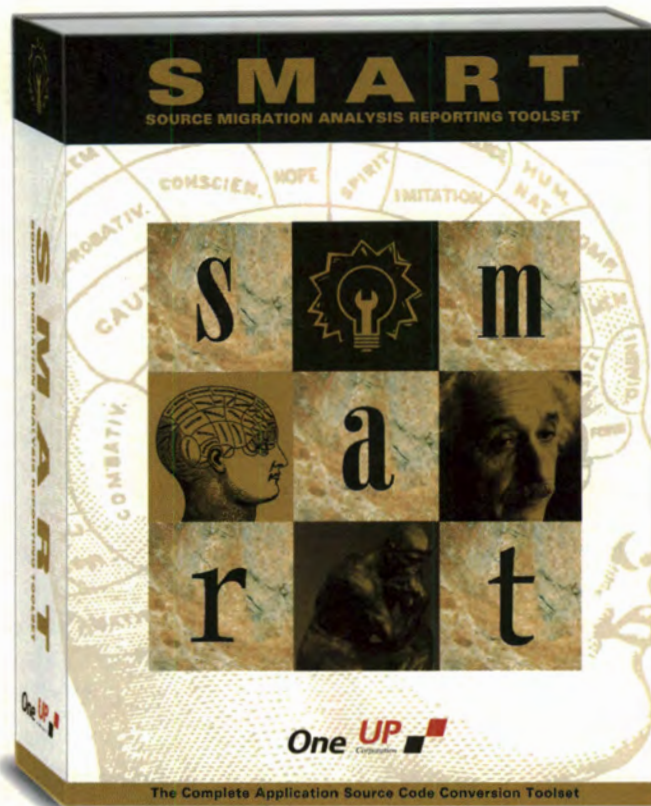
SOM_Scope BOOL      SOMLINK myn_wpPrintObject(MyNewObject *somSelf,
        PPRINTDEST pPrintDest,
        ULONG ulReserved)
{
    BOOL bSuccess = TRUE;
    /* MyNewObjectData *somThis = MyNewObjectGetData(somSelf); */
    /* MyNewObjectMethodDebug("MyNewObject","myn_wpPrintObject"); */

    /* DosExecPgm on the app after querying to see the name of the file */
    /* I represent (wpQueryRealName against myself). Pass that plus the flag */
    /* to the EXE (or call the entry point of the app DLL to print the file) */

```

Figure 2. Sample object's .C file (continued on page 28)

Converts Source Code To Native 32-Bit OS/2.



NO MIRRORS! NO KIDDING!

For more information, call 800-678-01UP or FAX 214-620-9626.



One UP
Corporation

1603 LBJ Freeway, Suite. 200 • Dallas, TX 75234

Circle Reader Service Number 11


```

    return (bSuccess);
}

/*
 *
 * OVERRIDE: wpSetup                                ( ) PRIVATE
 *                                                    (X) PUBLIC
 * DESCRIPTION:
 *
 *     All I need to do here is to
 *
 *     1) Set the object "type". This is the type associated with the object.
 *
 *     2) Modify the setup string to specify my iconfile name.
 *
 */

SOM_Scope BOOL    SOMLINK myn_wpSetup(MyNewObject *somSelf,
                                       PSZ pszSetupString)
{
    /* MyNewObjectData *somThis = MyNewObjectGetData(somSelf); */
    PSZ pszType;
    CHAR szPath[ CCHMAXPATH ];
    ULONG cb;
    /* MyNewObjectMethodDebug("MyNewObject","myn_wpSetup"); */

    cb = sizeof( szPath );
    if ( parent_wpSetup(somSelf,pszSetupString)
        && _wpQueryRealName( somSelf, szPath, &cb, TRUE ) ) {
        pszType = myn0_wpcIsQueryInstanceType(_somGetClass(somSelf) );
        if ( pszType ) {
            _wpSetType(somSelf, pszType, NULL);
            SetTypeEA( szPath, pszType );
        }
        return( TRUE );
    }

    return ( FALSE );
}

/*
 *
 * OVERRIDE: wpCreateAnother                          ( ) PRIVATE
 *                                                    (X) PUBLIC
 * DESCRIPTION:
 *
 *     Called whenever a new instance of this object class is
 *     created when the user selects "Create Another". Note that
 *     when the user selects drags one out from a template,
 *     we are not called here, but at wpCreateAnother. In here, as well as

```

Figure 2. Sample object's .C file (continued on page 29)



```
* in the override to wpCreateFromTemplate, we will call PutUpStyleSheetDlg
* to handle the style sheet work for us. We don't do it here so as not
* to have the same code in 2 places.
*
* Passed to me are a pointer to me, along with the folder in which the
* user wants to create the object. I'll call my parent class
* parent_wpCreateAnother to handle the meat of creation, I just want
* to give the user the choice of style sheet. (Kind of like when you
* subclass a window in PM.) (Make sure to pass the folder to
* PutUpStyleSheetDlg if you are going to do the actual data file creation
* from style sheet work there.)
*
*/

SOM_Scope WPObject * SOMLINK myn_wpCreateAnother(MyNewObject *somSelf,
        PSZ pszTitle,
        PSZ pszSetupEnv,
        WPFolder *Folder)
{
    /* MyNewObjectData *somThis = MyNewObjectGetData(somSelf); */
    MyNewObject *Object;
    MyNewObjectMethodDebug("MyNewObject", "myn_wpCreateAnother");

    Object = parent_wpCreateAnother(somSelf, pszTitle, pszSetupEnv, Folder);
    if ( Object ) {
        PutUpStyleSheetDlg();
    }

    return ( Object );
}

/*
*
* OVERRIDE: wpCreateFromTemplate ( ) PRIVATE
* (X) PUBLIC
* DESCRIPTION:
*
* Called whenever a new instance of this object class is
* created when the user drags one out from a template. Note that
* when the user selects "Create Another" from an existing instance,
* we are not called here, but at wpCreateAnother. In here, as well as
* in the override to wpCreateAnother, we will call PutUpStyleSheetDlg
* to handle the style sheet work for us. We don't do it here so as not
* to have the same code in 2 places.
*
* Passed to me are a pointer to me, along with the folder in which the
* user wants to create the object. I'll call my parent class
* parent_wpCreateFromTemplate to handle the meat of creation, I just want
* to give the user the choice of style sheet. (Kind of like when you
* subclass a window in PM) (Make sure to pass the folder to
```

Figure 2. Sample object's .C file (continued on page 30)


```

*   if you are going to do the actual data file creation
*   from style sheet work there.)
*
*/

SOM_Scope WPObject * SOMLINK myn_wpCreateFromTemplate(MyNewObject *somSelf,
    WPFolder *folder,
    BOOL fLock)
{
    /* MyNewObjectData *somThis = MyNewObjectGetData(somSelf); */
    MyNewObject *Object;
    MyNewObjectMethodDebug("MyNewObject", "myn_wpCreateFromTemplate");

    Object = parent_wpCreateFromTemplate(somSelf, folder, fLock);
    if ( Object ) {
        PutUpStyleSheetDlg();
    }

    return ( Object );
}

/*
*
*   OVERRIDE: wpclsQueryTitle                ( ) PRIVATE
*                                           (x) PUBLIC
*   DESCRIPTION:
*
*   Return the string "My Object Document" as the title of the class.
*
*/

#undef SOM_CurrentClass
#define SOM_CurrentClass SOMMeta
SOM_Scope PSZ SOMLINK myn0_wpclsQueryTitle(M_MyNewObject *somSelf)
{
    /* M_MyNewObjectData *somThis = M_MyNewObjectGetData(somSelf); */
    M_MyNewObjectMethodDebug("M_MyNewObject", "myn0_wpclsQueryTitle");

    return (myn0_wpclsQueryInstanceType( _somGetClass( somSelf) ));
}

/*
*
*   OVERRIDE: wpclsQueryInstanceType          ( ) PRIVATE
*                                           (x) PUBLIC
*   DESCRIPTION:
*
*   This will set up my file types for associations
*   This needs to match exactly the Type used in the
*   ASSOCTABLE

```

Figure 2. Sample object's .C file (continued on page 31)



```
*/
*/

SOM_Scope PSZ  SOMLINK myn0_wpclsQueryInstanceType(M_MyNewObject *somSelf)
{
    /* M_MyNewObjectData *somThis = M_MyNewObjectGetData(somSelf); */
    M_MyNewObjectMethodDebug("M_MyNewObject", "myn0_wpclsQueryInstanceType");

    return ("My Object Document");
}

/*
*
*  OVERRIDE: wpclsQueryInstanceFilter                ( ) PRIVATE
*                                                    (X) PUBLIC
*  DESCRIPTION:
*
*  This will set up the filter for the file types for associations
*  This needs to match exactly the file filter used in the
*  ASSOCTABLE
*
*/

#undef SOM_CurrentClass
#define SOM_CurrentClass SOMMeta
SOM_Scope PSZ  SOMLINK myn0_wpclsQueryInstanceFilter(M_MyNewObject *somSelf)
{
    /* M_MyNewObjectData *somThis = M_MyNewObjectGetData(somSelf); */
    M_MyNewObjectMethodDebug("M_MyNewObject", "myn0_wpclsQueryInstanceFilter");

    return ("*.MYN");
}

#undef SOM_CurrentClass

/*
* Force the ea on the file.  This is a work around for a
* workplace bug.  This is only a problem from wpSetup when the
* object is first created.
*/

BOOL SetTypeEA( PSZ pszFilePath, PSZ pszEAValue )
{
    USHORT cbValue;
    USHORT cbMem;
    USHORT rc;
    PFEA2LIST pfea2l;
    PSZ ptr, ptrEnd;
    PBYTE ptrValue, ptrCurrent;
```

Figure 2. Sample object's .C file (continued on page 32)


```

EAOP2 eaop;
PSZ pszEAName = ".TYPE";

ptr = pszEAValue;
cbValue = 0;
if ( ptr && *ptr ) {
    cbValue = strlen( ptr );
} /* end if */
if ( cbValue ) {
    cbValue += 2 * sizeof( USHORT );
} /* end if */
cbMem = (USHORT)sizeof( FEA2LIST ) + strlen( pszEAName ) + (USHORT)cbValue;
pfea2l =(PFEA2LIST) calloc( 1, (ULONG)cbMem );
if ( !pfea2l ) {
    return FALSE;
} /* end if */
memset((PCH)pfea2l,0,cbMem);
pfea2l->cbList = cbMem;
pfea2l->list[0].cbName = (BYTE)strlen( pszEAName );
pfea2l->list[0].cbValue = cbValue;
strcpy( pfea2l->list[0].szName, pszEAName );
if ( cbValue ) {
    ptr = pszEAValue;
    ptrValue = &pfea2l->list[0].szName[ strlen( pszEAName ) + 1 ];
    ((PUSHORT)ptrValue)[0] = EAT_ASCII;
    ((PUSHORT)ptrValue)[1] = cbValue - (USHORT)(2 * sizeof(USHORT));
    memcpy( &ptrValue[4], ptr, ((PUSHORT)ptrValue)[1] );
} /* end if */

eaop.fpGEA2List = NULL;
eaop.fpFEA2List = pfea2l;
eaop.oError      = 0;

rc = (USHORT)DosSetPathInfo( pszFilePath, 2L, &eaop, sizeof( eaop ), 0L );

free( (CHAR *)pfea2l );

return (USHORT)(!rc);
}

/* Private function: PutUpStyleSheetDlg
*
* This function is called in the overrides for both wpCreateAnother and
* wpCreateFromTemplate. This will put up the dialog that reads the
* "style sheet directory" for all the style sheets and puts them in
* a selection dialog for the user to choose from (don't forget to add a
* "non" option to the listbox.) You can then either copy the style sheet to
* the new data file here, or do it in the overrides. You should probably
* do it here to have the code in only one place.
*
* When I get to this function, I need to first put up a dialog asking

```

Figure 2. Sample object's .C file (continued on page 34)



Conferences,
October 3 - 7, 1994
Exhibition,
October 4 - 6, 1994

Washington Convention Center
Washington D.C.

Come See the Software and Application Development Industries Come Alive!

"The best speakers, the best demonstrations, and the best exhibition!" Terry McIntosh, Analyst - Chevron Corp.

SD '94 East delivers development solutions across the spectrum of tools from C++ to Powerbuilder.

- Client/Server Tools
- Object-Oriented Tools
- Rapid Application Development Tools
- Database Design and Modeling Tools
- CASE Tools
- GUI Front Ends
- Languages
- Configuration Management Tools
- Testing and Debugging Tools
- Porting Tools
- Multimedia Tools

SD '94 East provides powerful, real-world, real-time education in 4 jam-packed conferences.

SD was the first and continues to be best at bringing you practical, unbiased guidance in over 200 classes across 17 tracks.



Join us at SD '94 East as the industry comes alive...

...on the show floor, in the conference sessions and in a series of special events with key industry players.

- 4 Plenary sessions include:
Mitchell Kertzman, CEO, Powersoft
Christine Comaford, *PC Week*
Vaughan Merlyn, Partner, Ernst & Young
- Hours of free technical sessions by Microsoft, Powersoft, Intersolv and others
- The C++ Programming Superbowl



Tools and Techniques for Software and Application Developers

Call for a Brochure!

1-800-441-8826
415-905-2784
Internet: sd94east@mfi.com
FAX: 415-905-2222

vpCreateFromTemplate

The `vpCreateFromTemplate` method is called when (as the name implies) the user creates a new class instance by dragging one out from a template. Here you need to do the same as you did with `vpCreateAnother`. First, the parent method is called to actually create the instance. And then `PutUpStyleSheetDialog` is called to set the style sheet.

Of course, this entire style sheet business is optional. I added it to show you how you can call parent methods to provide the function of the parent class, while allowing you to override as much or as little of a parent method as you need (or just to make use of the parent's method and add to it rather than replace it).

CLASS METHODS

Class methods, as opposed to instance methods, are responded to by the class, rather than by the instances of the class. These are methods that all instances of this specific class obey because the class is doing the responding. These are the class method overrides for this sample object class. These are the basic methods almost any object you write will need to override. You will also notice the lines:

```
#undef SOM_CurrentClass
#define SOM_CurrentClass SOMMeta
```

before each class method. I alluded to this previously when I mentioned the specific lines the SOM emitters put before the class methods that are not around the instance methods. Because of these lines, do not put instance method overrides after the first class method override. When you add new instance method overrides to a .CSC file, the emitters will put the code at the end of the .C file. So, if you add new instance method overrides, be sure to move them

```
* the user which style sheet (s)he wants. To do this, I will look at my
* app defaults and figure out what directory the style sheets are in.
* I'll read the file list and give the user a choice of whatever style
* sheets are in there. This way a user can create his/her own style sheets
* and put them in this directory, and they will become part of AmiPro
* (nice feature). When the user selects the style sheet, create
* the physical file with the style sheet selected.
*
* You should probably pass the folder the new instance is being placed
* into to this function so you know where to create the datafile.
*
*/
PutUpStyleSheetDlg()
{

    DosBeep(329,300);
    DosBeep(392,350);
    DosBeep(329,75);
    DosBeep(349,300);
    DosBeep(392,325);

    return;
}
```

Figure 2. Sample object's .C file (continued from page 32)

above the first class method override before you compile.

vpClsQueryTitle

The `vpClsQueryTitle` method is called when an object wants to obtain the name or title of your class. This method is simple because it returns an ASCII string, which is the class name.

I coded the title so it matched the instance type. In this way, and as you will see in the following override, I only needed to return the string that is the title ("My Document Object") in the override for `vpClsQueryInstanceType`. Now in the override for `vpClsQueryTitle`, the title can be returned by invoking `vpClsQueryInstanceType` on the object itself (`myn0_vpClsQueryInstanceType`).

vpClsQueryInstanceType

The `vpClsQueryInstanceType` method is called when an object wants to

know the type of object this is. As explained previously, the type is `My Document Object`, which is what is being set in the .TYPE extended attribute and is the title of the class. To respond to a `vpClsQueryInstanceType` call, you need to return only the text string. The string must be the same as the title field of an `ASSOCTABLE` entry for the executable file. This linkage is how the shell links the .EXE and the object in its internal tables.

vpClsQueryInstanceFilter

The `vpClsQueryInstanceFilter` method is invoked when an object wants to know the file system filter for the file types for this type of object. This method, like the previous method override, helps the Shell map the executable file to the Shell's representation of the data file (object). In this case, the mapping is for the file name

extension or filter.

In the **ASSOCTABLE** statement in the resource file for the executable program, you specify not only the type name but also a file name extension or set of file name extensions. In the override for **wpclsQueryInstanceFilter**, you return the text string that is the file name extension filter (in this case, ***.MYN**).

PRIVATE FUNCTIONS

The private functions are only callable by the method overrides for this class. These are internal functions of the DLL; they will be specific to your class and cannot be called by anyone but you.

SetTypeEl

This function sets the **.TYPE** extended attribute for the file name passed to it.

PutUpStyleSheetDlg

Although this function does nothing but play a tune, its purpose is to perform the initialization you want during the initialization of a new instance of the object class.

TYING IT TOGETHER

Some of what I have covered in this column may not seem immediately obvious. You will understand it gradually as you work with Workplace Shell code.

When you look at this object as an entity, first notice that any

object entity has several interfaces to the Shell. For example, you can "link" applications to objects using several mechanisms. The first is the **ASSOCTABLE** in the executable. When the executable program is first awakened by the Shell, the **ASSOCTABLE** is read and associations are built. These associations hook the executable program with a file extension or filter and a type.

These items (the filter and type) are two of the class methods that are overridden. The values returned in these overrides must match the values you place in the **ASSOCTABLE**. In this way, the Shell can access the object and invoke the executable no matter how the user accesses it (from the program object, from the document object, and so on).

Also notice that there is no explicit code for the icon handling for the object class. If you use an **ASSOCTABLE** statement and specify an icon, the Shell will automatically pick up the icon from that; you do not need specific code in the object class definition for custom icons.

By using the small set of overrides in **wpSetup**, **wpclsQueryInstanceType**, **wpclsQueryInstanceFilter**, and **wpclsQueryTitle**, along with an **ASSOCTABLE** in the executable, you can provide the fundamental functions of a Workplace Shell object.

NOW IT'S UP TO YOU

Once you have completed these functions, you can customize the class's behaviors with overrides, such as **wpPrintObject** and others.

Carefully examine everything you do in your object class, because the shell-defined classes are powerful and versatile. You will be tempted to concentrate on overrides, but carefully examine the already-defined methods. There may be some function you can use rather than rewrite. Whenever possible, use the class's code rather than write it yourself. Refer to the example in **wpCreateFromTemplate**; notice that it calls the parent method to create the instance, then, before returning to the caller, performs the customization work. This concept can save you a lot of time and money.

Use the sample object as a skeleton. If you combine it with the **MAKE** file from the my last column, you'll be all set to write Workplace Shells objects.

David Reich has been with the IBM OS/2 development team since 1987. He has worked on many parts of the system, supported customers and application developers, and traveled the world giving seminars and teaching OS/2. He is the author of *Designing OS/2 Applications*, published by John Wiley and Sons. He can be reached on CompuServe at 76711,632 or via the Internet at speedracer@vnet.ibm.com.





Object-Oriented Programming

*The emerging OpenDoc technology promises a cross-platform, language-independent, LAN-supporting standard for the exchange of compound documents among a variety of applications. We'll have more to say about OpenDoc and other OS/2 object programming topics in our September/October issue. For now, here's an introduction to get you started. By **ROBERT L. TYCAST***

Component Software for the Masses: OpenDoc is Here

Several years ago, the electronics industry was producing console stereos. These units were sold more like furniture than electronic devices. One large cabinet came complete with television, record player, speakers, and AM/FM radio.

Consumer choices were limited. If you didn't like the speakers, you had to live with them or buy a different model, but then maybe the record player wouldn't be up to your standards. There was no way to mix and match the pieces of a system to match your needs, tastes, or budget. And if that wasn't bad enough, when new technology arrived, you couldn't just add it to the box. You had to scrap what you had and buy a complete new package.

Today's consumers would never tolerate that. They are accustomed to picking and choosing their compact disc players, speakers, tuners, VCRs, and video monitors to build their personal home entertainment systems. Shoppers have complete freedom to choose from a large number of products produced by different manufacturers using an ever-growing number of new technologies. They need not worry that their home system will be instantly obsolete because of a new product they might want to buy next year. They are safe in

assuming that the producers of consumer electronics will make sure that next year's products will plug into the home system purchased today.

This is ideal for both producers and consumers. Consumers have more flexibility and choices, and producers have a bigger market in which to sell their technology. Upgrades to existing systems are easy, so a modest advance in technology can be parlayed into a successful moneymaker. New components can be marketed aggressively since consumers no longer wait until their console systems "smoke" themselves into oblivion, requiring the consumers to throw them away and buy new ones.

Today's software industry is in many ways comparable to the electronics industry of yesteryear.

Applications are monolithic, containing every possible function users might need (whether or not they really want them), locking them into a fixed way of working. New software technology requires upgrades or replacements. Finding a cooperative set of applications means buying into a suite of products, usually from a single vendor. There is no way to mix and match applications, no way to add new technology to an existing suite without a major update, and no easy way for the user to build a custom system.



The change in home electronics from console to component systems was revolutionary. The coming changes in the software industry ushering in component software will be no less so. And the first step in that revolution is called OpenDoc.

OPEN SYSTEMS, OPEN STANDARDS

OpenDoc is a collection of technologies donated to the industry by Apple, IBM, and WordPerfect, enabling the building of compound documents that can be exchanged, viewed, and edited between multiple platforms.

Apple contributed the basic compound document technology of OpenDoc. Apple also created Bento, a technology that addresses the problems inherent in storing compound documents, and Open Scripting Architecture (OSA), which makes interoperability between documents and parts of documents a breeze.

IBM donated its System Object Model (SOM) technology, which is a Common Object Request Broker Architecture (CORBA) compliant technology, enabling documents to be built as objects and distributed across a network. Finally, WordPerfect is contributing technology that will enable OpenDoc documents to work with Microsoft's OLE2 servers and containers.

The technologies will be owned and distributed by a consortium, Component Integration Laboratories (CILabs). Modeled after the X Window System Consortium, CILabs will distribute the OpenDoc technologies, promote industry standards, and provide certification of OpenDoc compliance for software products, ensuring that shrink-wrapped packages work together from the start.

In addition, Apple is providing a reference implementation of OpenDoc on its Macintosh platform. WordPerfect

is creating one for MS-Windows, and IBM is writing OpenDoc implementations for AIX and OS/2.

PARTS AND PART HANDLERS

Key to the compound document architecture is the concept of a part. Parts are the individual components used to create a document. The most common one is the text part. It provides the ability to embed and edit text into a document. Other part-types may include graphical drawing parts; spreadsheet parts; multimedia parts, such as video and audio; spell-check parts; and so on. The list goes on and on because the OpenDoc architecture is designed to support all types of data—even data types not invented yet!

How is this done? The basic programming unit is the part handler, and every supported data type has one. One nice thing about part handlers is that they need only concern themselves with the bit of functionality that they provide. Developers of today's monolithic applications must deal with every aspect of functionality provided. OpenDoc part writers, on the other hand, can concentrate on the area of specialty their part provides, unconcerned about other ancillary functions that users might want. After all, if users want to add other functions to the application later, they only have to add some part handlers.

STORING THE DOCUMENT

Once users have assembled documents with a diverse set of parts, they will want to store and exchange them with other users. Handling a document made up of multiple parts and data types would be less than trivial if it were not for OpenDoc's Bento technology.

Bento is designed to provide storage containers for multiple data types in an efficient and flexible way. As a docu-

ment is modified, Bento creates new drafts. This archiving of the document is very storage-efficient, since only the changes from the previous draft are stored.

The Bento system provides an easy way to create links to other parts of a document or even outside of the document. You can create a piece of a document once and link it to any number of other documents, even when they are on diverse network nodes and platforms. This is important since today's and tomorrow's computing environments are distributed, and the networks on which parts and documents will be found will most certainly be heterogeneous.

SCRIPTING, EVENTS, AND RECORDABILITY

Scripting, events, and recordability are all features supported by OSA-compliant parts. OSA defines the way in which documents and parts communicate with each other, and OSA provides a standard for part handlers that guarantees interoperability. This standard is called the event registry. It contains lists of command verbs that applications use and the objects on which they operate.

The registry gives precise definitions on how to encode the verbs and objects. The verbs are further organized into suites according to the type of applications that typically use them.

Some examples of the standard suites are:

- Required: the base set of functions that all compliant parts will need to support
- Core: defines file and edit functions and basic scripting
- Text: defines basic text editing and formatting
- Table: defines commands used for table editing and formatting.

Once OSA support has been incorporated into a part, script-

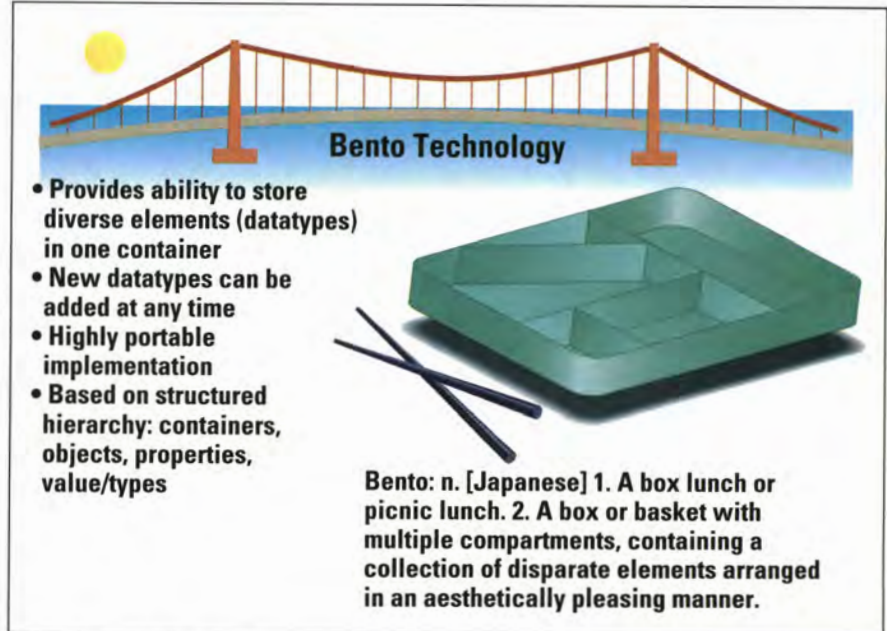


Figure 1. OpenDoc storage

ability, recordability, alternate input, and other features become a reality. To get a sense of the power of OSA, here's an example of how a typical OSA scripting component might read:

```
tell application "My editor"
  select word 1 of paragraph 2
```

```
cut selection
end tell
```

The OSA script closely resembles natural language. In fact, OSA makes it possible to create language-independent scripting components. Since the underlying mechanism is via OpenEvents and

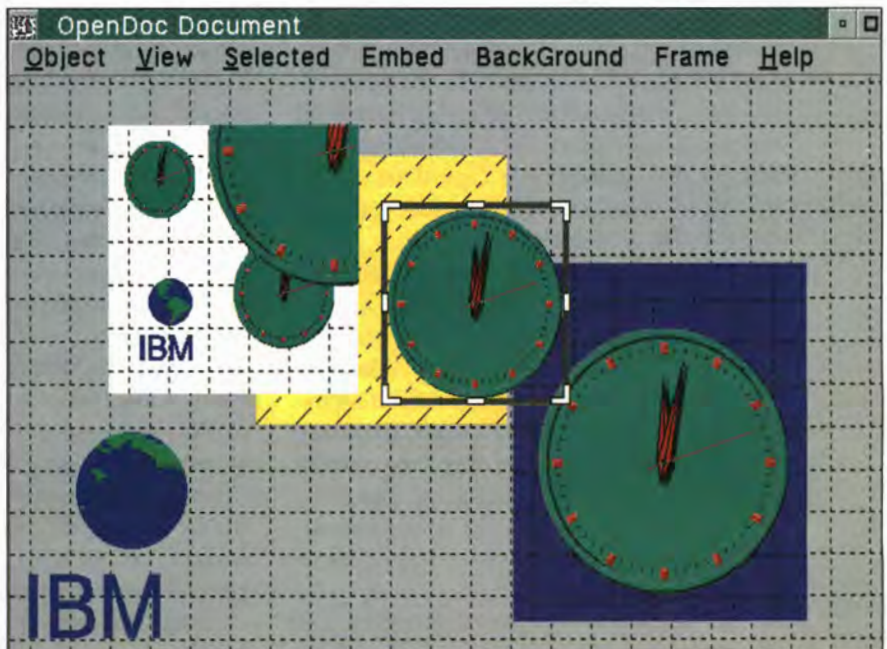


Figure 2. OpenDoc for OS/2 is on its way

**After November 4, 1994, there will
be two kinds of OS/2 developers.
Those who went to ColoradOS/2.
And those who didn't.**

**ColoradOS/2
October 30-November 4, 1994
Cheyenne Mountain Conference Resort,
Colorado Springs, CO**



ColoradOS/2. The definitive conference for OS/2 developers, in the shadow of Pikes Peak.

Those who attend will get in on the very first in-depth analysis, discussion and demonstration of exciting new technologies like OpenDoc, WorkPlace OS and the first new object-oriented products from the Taligent Frameworks partnership. Plus the latest on SOM/DSOM and WPS.

Those who attend will learn about object-oriented design directly from Grady Booch, the man who wrote the book on it.

Those who attend will hear about the REXX language directly from its creator, IBM Fellow Mike Cowlshaw. They'll get the inside story directly from IBM Director of Object Technology Products Cliff Reeves. And they'll talk in-depth about internationalization and DBCS.

And those who don't, won't.

ColoradOS/2. Miss it, and you'll miss a lot. Make it, and you've got it made.

Plan now to attend ColoradOS/2.

For reservations, call the Cheyenne Mountain Conference Resort at (800) 648-5717 or (719) 576-5003. For more information, call (800) 481-3389 or (719) 481-3389.



IBM, OS/2 and other IBM products are trademarks or registered trademarks of International Business Machines Corporation. Other product names are trademarks or registered trademarks of their respective companies.

Circle Reader Service Number 12



object specifiers, a script editor could translate the preceding script to:

```
utilise l'application "My editor"  
  selectionne mot 1 du paragraphe 2  
  coupe selection  
fin de utilise
```

with the simple click on a "Français" radio button. (This example uses a format typical of the AppleScript language).

Since applications can respond to OSAEvents, it's easy to monitor event traffic and record it for later playback. Another advantage of factoring a part to be driven by OSAEvents is that any component—the user interface, the script language, voice or pen input—can generate an OSAEvent and drive the application.

BRINGING IT ALL TOGETHER— SYSTEM OBJECT MODEL

The SOM technology contributed by IBM sets the foundation for OpenDoc firmly on new object technology. SOM provides a CORBA-compliant way for parts to create objects that are platform- and language-independent and that can work in a distributed environment. This object "glue" is an essential ingredient to the CILabs strategy of

supplying industry-standard object technology.

The changeover to component software will be as profound a shift for users and software producers as the shift to third-generation languages like FORTRAN, C, and PASCAL was in its day. A key part of IBM's overall strategy is to exploit its software technology. OpenDoc builds on the lead that SOM represented in the objects world. It is an important step to taking OS/2 to new levels of functionality and flexibility.

WHERE TO FIND OUT MORE

This article has only touched the surface of OpenDoc technologies. Subsequent articles will go into more depth about how the architecture fits together and how

developers actually write part handlers. Look for this series to continue in the September/October 1994 issue of *OS/2 Developer*.

Robert L. Tycast, IBM Entry Systems Division, Boca Raton, Fla., is an advisory programmer for the Advanced Presentation Manager Development group. He joined IBM Corp. in 1989 from Digital Equipment Corp., where he served in a number of software development roles, including providing software support in the U.S., Latin America, and Europe. His project experience over the last 13 years includes X11, AI Technology (LISP and OPS5 support), and technical workstations (VMS and ULTRIX). Tycast has a B.S. from Massachusetts Institute of Technology and has done graduate work in MIT's computer science department.

REFERENCES

In the meantime, you can contact CILabs directly for additional (preliminary) information about OpenDoc:

Component Integration Laboratories
688 Fourth Ave.
San Francisco, CA 94118
(415) 750-8352 (voice)
(415) 757-4829 (fax)
Internet: jed@cil.org

OnCmd[®] xBase for OS/2 ... A Winning Combination!

OnCmd harnesses the power and the speed of OS/2, enabling you to develop new applications or convert existing xBase applications, such as those developed in FoxPro[®], Clipper[®], and dBase[®].

Preserving Your xBase Investment

OnCmd can quickly convert your xBase applications to run directly in OS/2's 32 bit Presentation Manager with few modifications. This eliminates development time and preserves your current xBase investment. Text applications are automatically converted to a

native Presentation Manager application that is both multi-user and network capable.

With simple code changes, you can add buttons, check boxes, and drop-down menus to give your application more functionality.

Applications Development

OnCmd's user-friendly development environment makes developing new applications easy and efficient. The OnCmd screen painter brings point and click simplicity to screen designs,

allowing you to develop the layouts you need, while preserving your control over application flow. Support for "event-driven" programming and the use of extensible Dynamic Link Libraries (DLL) offers even greater power to your database.

Client/Server Ready

OnCmd allows you to create a client server environment without the need for a dedicated server system. You can evolve into the world of client/server at your own speed and at a very low cost, allowing you to take advantage of the improved performance and application stability of client/server.

Multi-User Networking

OnCmd automatically enforces multi-user database integrity through the network with no code changes. File and record

locking can be added to your application if higher level transaction integrity is required. You can add message passing to create co-operative or client/server relationships in your applications, including direct messaging to your "C" programs.

A Performance Winner

OnCmd is a performance winner! No windows emulation overhead. Disk requirements less than 1MB. Installation under 5 minutes. In addition, it typically indexes large databases twice as fast as the competition in half the disk space. For more benchmark details, call or email us.

*Go for the gold.
Order your copy today!*



ON-LINE

ON-LINE DATA
5 Hill Street, P.O. Box 65
Kitchener, Ontario Canada N2G 3X4
Tel: (519) 579-3930 Fax: (519) 579-2130
CompuServe: 70022,104
Internet: oncmd@onlinedata.com

* Prices are in US funds.
Add \$249 per 10 user blocks (Regular \$498).
Offer expires August 30/94.



1-800-265-2455

Product also available from: Indelible Blue Inc. 1-800-776-8284 (USA) 1-800-672-4255 (CDN.) Programmer's Paradise 1-800-445-7899 The Software Store (IBM) 1-800-342-6672
Online Data recognizes the following trademarks: FoxPro (Microsoft Corporation), dBase (Borland International Inc.), Clipper (Computer Associates International Inc.), OnCmd (Online Data).

Circle Reader Service Number 13



Pen Systems

In this article, we'll show you how to use the pen as an input device, and even make it talk back to you!

By VERA DULANEY and KEVIN LEE

The Talking Pen: Coding a Pen/Audio Application

Napoleon once said, "The pen is mightier than the sword," but we Pen for OS/2 developers prefer, "The pen is mightier than the mouse." The pen gives the user a much more intuitive pointing device than the mouse. The pen also gives more information about its movement than the mouse can.

PEN VS. MOUSE

When you move a mouse, OS/2's Presentation Manager handles the position of the pointer and knows when the mouse button is pressed or released, but any activity between the start and end points is ignored. Pen for OS/2, on the other hand, follows the pen's movement across a touch-sensitive screen or digitizer and keeps all the point locations in a buffer. The collection of points between the position where the pen first touches the surface and the position where it is lifted up is called a stroke.

INKING

Pen for OS/2 provides messages and functions for a user application to process pen movement and produce the ink, which is the visible trail of the pen moving across the screen. Users can do their own inking of pen movement if they want a specific color or width of the ink or let Pen for OS/2 ink in the

default color. Also, users can collect and manage pen movement or let Pen for OS/2 build a stroke buffer that stores all pen positions. When the buffer is built by Pen for OS/2, it is built in real time, and users are notified of its availability. If users want their own format of stroke buffer or do not need the data in the stroke buffer, the buffer is not built by Pen for OS/2.

UNLIKE THE SPACE SHUTTLE, TOUCHDOWN COMES BEFORE LIFTOFF!

There are four messages used to process a stroke:

WM_TOUCHDOWN	Makes contact with surface
WM_LIFTOFF	Breaks contact with surface
WM_MOUSEMOVE	Moves pen
WM_SENSOR_MOVE	Moves pen.

All pointing device messages are sent to the application; Pen for OS/2 treats pen messages as mouse messages. The WM_TOUCHDOWN message is sent first to the window under the point of contact. There are three types of return codes: the delay between the touch-down point and the mouse-button-down event message, an option to override the default inking's color and width, and options to receive coordinate data at full bandwidth, which sets the hardware sampling rate.



The application must call `WinSetCapture` function while the `WM_TOUCHDOWN` message is being processed. There are several ways to handle inking and the stroke buffer, described in the *Pen for OS/2 Programming Guide*.

Pen movement messages are sent to the application while the pen is in contact with the surface. Depending on the return code of `WM_TOUCHDOWN`, the `WM_SENSOR_MOVE` or `WM_MOUSEMOVE` message will be sent. The application can ink the stroke or perform other operations, such as erasing, as the messages are received at different surface locations.

The `WM_LIFTOFF` message is sent to the application at the end of the stroke. Call `WinSetCapture` with `NULLHANDLE` to release the pointing device message capture. If `TDN_NO_STROKE` is returned to the `WM_TOUCHDOWN` message, the application can access the stroke buffer through the `WrtQueryStrokeData` function. To retrieve the data, the application must first query the size of the buffer to allocate sufficient memory to store it. Stroke data can be provided in either screen coordinate or window coordinate format. The application can replay retrieved stroke data later on the screen.

If `TDN_INFINITE_PAUSE` is returned to the `WM_TOUCHDOWN` message, Pen for OS/2 does the inking of the stroke and provides a stroke buffer. As points are entered into the stroke buffer, they are connected to the previous points and displayed on the screen to give the appearance of ink flowing from the point of the pointing device. The inking is done in such a way as to not interfere with the underlying graphical data. At the end of the stroke, the `WM_LIFTOFF` message will be sent to the application and a stroke buffer is available. The de-inking of the stroke will be done upon return from the `WM_LIFTOFF` message.

Due to the high rate of movement messages determined by the hardware sample rate, each message must be processed in a short period of time. If

the application falls behind, it will still receive all the messages, but the processing of the points will appear slow compared to the pen movement.

Pen for OS/2 currently limits the size of the stroke buffer to 1,000 points or 10 seconds of activity. It checks each stroke; if it recognizes one as a predefined gesture, it executes an appropriate command. But the stroke can be used in other ways, such as signature verification. Pen for OS/2 records not only the signature but also the relative speed of writing each letter in the name and even the timing of dotting the Is and crossing the Ts.

Our example combines the use of pen and multimedia to save a pen stroke and audio response in a Presentation Manager window. We used the audio recording and playback feature of `MMPM/2`, the IBM multimedia system. This sample program (`MMINK`) is available in the current Pen for OS/2 Toolkit, on the Developer Connection CD-ROM. The voice attachment dialog box provides buttons to start recording, stop recording, and play back the recorded audio. While the program is recording the audio, it saves the stroke data through the `WM_STROKE` message. The key point is to store the stroke and audio data with an internal timer so they can be played back synchronously.

This is an example of an application that could use the synchronous annotation of voice and ink data on a document. Suppose an office manager is reading a document displayed as a bitmap in a Presentation Manager window and wants to modify the document. The manager can use a pen to circle an area to modify and record voice instructions at the same time. Later, the secretary can replay the ink and audio annotation on the document and modify it. If the manager's and secretary's computers are connected by a network, all document, stroke, and voice data can be sent between them directly by electronic mail.


```

MRESULT EXPENTRY InkWinProc(HWND hWnd, ULONG msg, MPARAM mp1, MPARAM mp2)
{
    HWND      hwnd;          /* My window handle          */
    HPS        hps;          /* Handle to presentation space */
    POINTL     points;       /* Point structure for saving last point */
    SIZEL      OrgWindowSize; /* Size of window at time of stroke input */
    LONG       CurrentColor; /* Current ink color          */
    LONG       NumSegments;  /* Number of graphic segments in PS */
    ULONG      buflen;
    PSTROKEDATA pStrokeData;
    APIRET      rc;
    static USHORT PointCounter = 0;
    static POINTL pPointU8e;
    static BOOL StrokeBufferOverrun = FALSE;
    RECTL       rcl;
    WRTEVENTDATA EventData;

    switch(msg)
    {
        /******
        * Assume the initial setup is done, for example, creation of the *
        * presentation space.                                           *
        *****/
        case WM_TOUCHDOWN: /******
            * Capture mouse messages, move to the point where the pen touches *
            * the workpad to ink.                                           *
            *****/ {
                GpiOpenSegment(hps, ++NumSegments );
                WinQueryWindowRect(hWnd, &rcl); OrgWindowSize.cx = rcl.xRight; OrgWindowSize.cy = rcl.yTop;
                PointCounter = 0;
                WinSetCapture ( HWND_DESKTOP, hWnd );

                /******
                * Note that the (SHORT) cast is required in order to maintain *
                * sign. *
                *****/
                / points.x = (SHORT) SHORT1FROMMP ( mp1 );
                points.y = (SHORT) SHORT2FROMMP ( mp1 );
                GpiMove (hps, &points );

                /******
                * TDN_INFINITE: Inform Pen for OS/2 that the mouse button down *
                * is not required. *
                * TDN_HIFREQ_MOUSEMOVE: Inform Pen for OS/2 to report the points *
                * as mouse moves but at the sensor sample rate. *
                * TDN_NO_INK_STROKE: Inform Pen for OS/2 not to ink the points *
                * since we will be doing our inking. *
                *****/
                / return ( (MRESULT) ( TDN_INFINITE
                    TDN_HIFREQ_MOUSEMOVE ]
                    TDN_NO_INK_STROKE ));
                break;
            }

        case WM_MOUSEMOVE:
            /******
            * For this message, the inking is done to show the stroke. *
            *****/

```

Figure 1. Sample program for handling pen strokes (continued on page 46)

The Future of Business Computing is Here Today!

Introducing

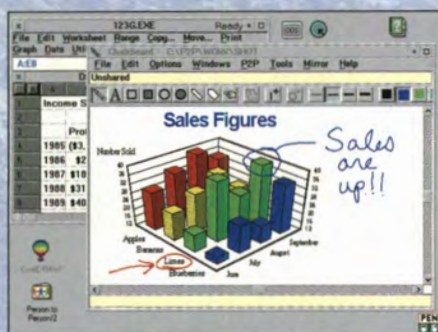
PenDirect™ for OS/2™

WORK MORE PRODUCTIVELY — Direct on-screen input means quicker and easier interaction with your computer. PenDirect is the most advanced light pen solution available today for OS/2.

WORK FASTER — PenDirect for OS/2 is *very* fast. Click, point, drag and draw with greater speed and precision than you can get from any other pointing device, plus a light pen won't fight with you for desk space. And with PenDirect's easy-to-install hardware (ISA, MCA or optional external) and solid award-winning software, you'll be up and running in just minutes.

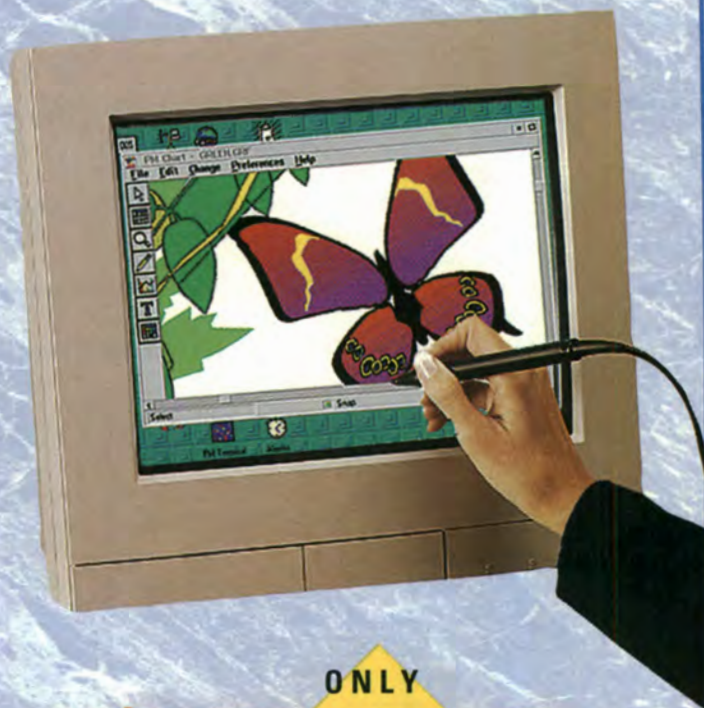
WORK SMARTER — PenDirect works great with all of your OS/2 software. Enhance graphics programs and bring desktop conferencing to life. Effortlessly navigate large spreadsheets, and get real light pen support for your 3270 apps. PenDirect fully supports all OS/2 2.x, Windows and DOS apps — at any screen resolution. Everything you need is included, no special monitor is required, and PenDirect lets you continue to use your mouse.

WORK IN COMFORT — PenDirect's one-to-one cursor interaction means long-term computer use that's simple and effortless. The pen's precise and quick input maximizes cursor control to eliminate "mouse dyslexia." You'll find your natural hand/eye coordination beats fumbling with an awkward mouse any day. Light pens also eliminate the tedious wrist movements associated with mice that cause fatigue and repetitive strain injuries.



Bring pen computing functionality to desktop conferencing software, like IBM's Person to Person.

WORK SAFER — PenDirect for OS/2 comes from FTG Data Systems, recognized as the world leader in light pen technology for PCs. FTG sells reliable American-made products backed with the best warranty in the industry, outstanding toll-free customer support and an unconditional no-risk, no-hassle evaluation/return policy. With over 100,000 pens installed, FTG is your safe choice and your best choice.



ONLY
\$298

limited-time introductory offer
(regularly \$398)

For a free 30-day evaluation unit,
or more information, call:

800-962-3900

714-995-3900 • FAX 714-995-3989

**FTG DATA
SYSTEMS**

8381 Katella Avenue
Stanton, California 90680




```

{
/*****
* This release of Pen for OS/2 has a maximum limit to the size *
* of the stroke buffer. It is limited to 1024 points or eight *
* seconds per stroke. If the maximum size of the stroke buffer *
* is reached, then inform the user via a tone. *
*****/
if ( StrokeBufferOverrun )
{
    DosBeep ( 4000, 16 );
    return ( (MRESULT) TRUE );
};
/*****
* Get the event data to see if the stroke buffer has been *
* been filled. *
*****/
EventData.cbStructSize = sizeof(EventData);
WrtQueryEventData ( &EventData );
if ( EventData.flEventStatus & WRT_BUFFER_OVERRUN )
{
    DosBeep ( 4000, 16 );
/*****
* Force the inking for all of the remaining points. *
*****/
    GpiBeginPath(hps, 1);
    GpiPolyLine(hps, PointCounter, pPoint);
    GpiEndPath(hps);
    GpiStrokePath(hps, 1, 0);
    GpiCloseSegment(hps);
    StrokeBufferOverrun = TRUE; return ( (MRESULT) TRUE );
};
/*****
* Keep the (x,y) coordinate in a buffer of 8 entries. *
* Note that the (SHORT) cast is required in order to maintain *
* sign. *
*****/
pPointUPointCountere.x = (SHORT) SHORT1FROMMP( mp1 ); pPointUPointCountere.y = (SHORT) SHORT2FROMMP( mp1 );
PointCounter++;
if (PointCounter == 8)
{
    GpiBeginPath(hps, 1);
    GpiPolyLine(hps, 8L, pPoint);
    GpiEndPath(hps);
    GpiStrokePath(hps, 1, 0);
    PointCounter = 0;
}
return ( (MRESULT) TRUE );
break;
}
case WM_LIFTOFF:
/*****
* Stroke ends. Collect stroke end time, No more mouse message *
* capture, Get the stroke data. *
*****/ {
    WinSetCapture ( HWND_DESKTOP, NULLHANDLE );

```

Figure 1. Sample program for handling pen strokes (continued on page 47)



```

/*****
 * If the buffer was overrun, then the points have already been
 * displayed.
 *****/
if ( !StrokeBufferOverrun )
{
/*****
 * Force the inking for all of the remaining points.
 * Note that the (SHORT) cast is required in order to maintain
 * sign.
 *****/
pPointUPointCountere.x = (SHORT) SHORT1FROMMP( mp1 ); pPointUPointCountere.y = (SHORT) SHORT2FROMMP( mp1 );
PointCounter++;
    GpiBeginPath(hps, 1);
    GpiPolyLine(hps, PointCounter, pPoint);
    GpiEndPath(hps);
    GpiStrokePath(hps, 1, 0);
    GpiCloseSegment(hps);
};
/*****
 * Get the size of the stroke buffer.
 */

```

Figure 1. Sample program for handling pen strokes (continued on page 48)

OS/2 DEVELOPER BACK ISSUES

LIMITED NUMBER OF ISSUES AVAILABLE

The following Back issues of OS/2 DEVELOPER are available for \$9.95 each. (Add \$2.50 for shipping)

YES, PLEASE SEND ME:

- ☐ Winter 1990, DOS to OS/2 Conversion
- ☐ Fall 1990, Multimedia & Graphics
- ☐ Fall 1991, Computer Integrated Manufacturing
- ☒ Winter 1992 **SOLD OUT**
- ☐ Spring 1992, 32-bit Tools
- ☒ Summer 1992 **SOLD OUT**
- ☒ Fall 1992 **SOLD OUT**
- ☐ Winter 1993, Migration Strategies
- ☒ Spring 1993 **SOLD OUT**
- ☐ July/August 1993, Taligent Spotlight
- ☐ Sept/Oct 1993, Networking with OS/2
- ☐ Nov/Dec 1993, SOM/DSOM

Subscribe to OS/2 DEVELOPER for one year and save 33% off the cover price.

YES, PLEASE SEND ME:

- ☐ 1 year (six issues) – \$39.95 in the U.S.

Canada, Mexico, and international surface mail—add \$16 per year
International air mail—add \$30 per year

Please check the appropriate boxes on this page and complete the form below, and mail this page to:

OS/2 DEVELOPER

P.O. Box 1079
Skokie, IL 60076-9772
or fax: 708-647-0537
phone: 800-926-8672 (800-WANT-OS2)

Please Print:

NAME _____

TITLE _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

☐ Check enclosed

☐ Charge my credit card:

☐ Visa

☐ Mastercard

☐ American Express

CARD NUMBER _____

EXPIRATION DATE _____

SIGNATURE _____


```

*****/
buflen = 0;
pStrokeData = NULL;
rc = WrtQueryStrokeData((PBYTE) pStrokeData, &buflen, hWnd, QSD_SCALE, 0, 0, 0);
if (rc != 0) !buflen)
    return (MRESULT) TRUE;
/*****
 * Allocate memory for the stroke buffer.  *
*****/
pStrokeData = (PSTROKEDATA) malloc(buflen);
if ( !pStrokeData )
{
    return ( (MRESULT) TRUE );
};
/*****
 * Get the stroke data  *
*****/
pStrokeData->cbStructSize = sizeof ( STROKEDATA );
if ( WrtQueryStrokeData ( (PBYTE) pStrokeData,

```

Figure 1. Sample program for handling pen strokes (continued on page 49)

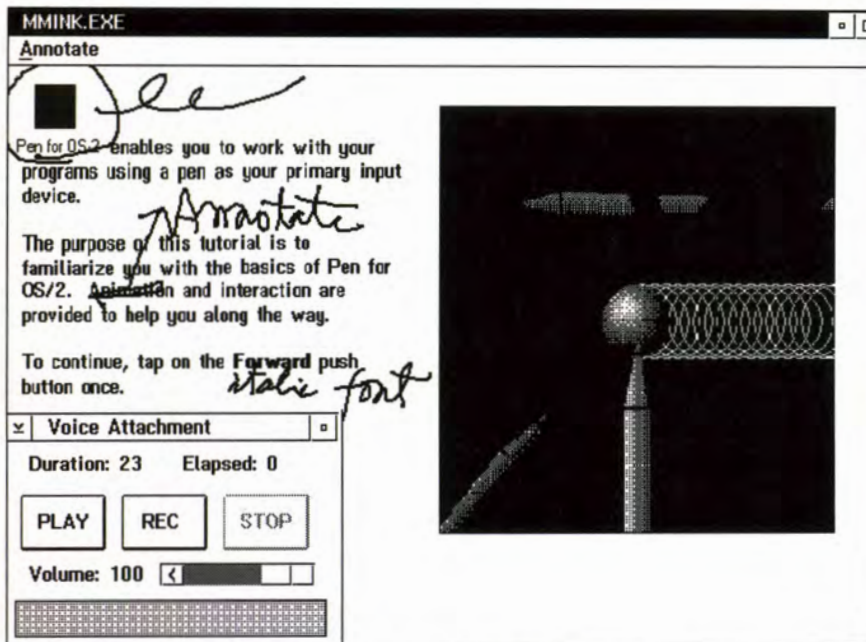


Figure 2. Sample screen with corrections made by both ink data and voice annotation

Figure 2 is an example of what it would look like if the manager wanted to remove the Pen for OS/2 icon, change "Animation" to "Annotation," and change the bold font of Forward to italic.

As shown, the pen is capable of more functions than the mouse. If the stroke is recognized as a predefined gesture, Pen for OS/2 exe-

cutes the command that is preassigned to the gesture. The pen can be used for handwriting control, which enables the Pen for OS/2 user to input handwritten text. The text is recognized and converted to alphabetic and numeric characters.

Pen for OS/2 displays virtual keyboards. This is useful particularly for a keyboardless system or

for applications that require a specialized keyboard. Several applications that take advantage of the pen functions are already in use; for example, parcel delivery systems. More applications will be available as more pen-based systems are developed.

Vera Dulaney, IBM Personal Systems Programming, Boca Raton, Fla., is manager of the MVP Pen and Speech Design and Development department. She has been a member of the OS/2 development organization.

Kevin Lee, IBM Personal Systems Programming, Boca Raton, Fla., is a member of the MVP Pen and Speech Design and Development department. He joined IBM in 1983. Lee has worked on compiler development, transaction processing systems, language theory, and OS/2.

Complete code blocks shown in this article are available on CompuServe's OS2DF2 Forum, OS/2 Developer section. Download file PENCOD.ZIP



```

&bufLen,
                                hwnd,
QSD_SCALE,
                                0,
                                0,
                                0 ) )
    return ( (MRESULT) LO_STROKE_PROCESSED );
/*****
 * Append the current stroke data to doubly linked list for later *
 * use.
 *****/
/*****
 * Use existing structure entries for the stroke to hold the *
 * specific data of the current stroke, for ex., touch down time *
 * and lift off time. *
 *****/
    StrokeBufferOverflow = FALSE;
    return ( (MRESULT) LO_STROKE_PROCESSED );
}
}
}

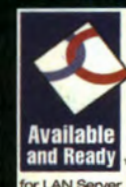
```

Figure 1. Sample program for handling pen strokes (continued from page 48)

Now You Can Easily Backup, Restore, Migrate and Manage Your Workplace Shell Desktop With ...



DeskMan/2



DeskMan/2™ is the first and only tool designed to manage the Workplace Shell™!

DeskMan/2 for OS/2 is unmatched in its ability to fully exploit the power and versatility of OS/2! Managers can use its security features to protect access to windows or objects. Administrators can easily and quickly standardize a department's desktops for maximum job efficiency. Individual users can quickly and easily migrate icons, objects and desktops between office and home machines. Or use **DeskMan/2** to backup and completely restore a custom desktop.

Incorporating the powerful **VUEMan/2™** virtual desktop & window manager; the potent new WPS snapshot facility; and more, **DeskMan/2** is packed with features!

"I installed OS/2® 2.1 golden code, and **DeskMan/2** saved me at least 4 hours of tedious labor. I can't imagine being without it."

David Barnes - Senior Staff member - IBM's OS/2 Executive Briefing Center

- | | |
|---|--|
| ☒ API & GUI for local & remote access | ☒ Change multiple folders or icons all at once -- just drag and drop |
| ☒ CID enabled & LAD/2 supported | ☒ Full administrative control of all features |
| ☒ Delete "undeletable" objects | ☒ Restore, save, migrate any/all objects |
| ☒ Prevent the deletion of any object | ☒ Total management of Workplace Shell |
| ☒ Query any object for settings | ☒ And much more ... |
| ☒ Secure individual windows | ☒ Call DevTech for current info! |
| ☒ Secure groups of windows | |

DevTech

Development Technologies, Inc.

Software Development & Technology Transfer

308 Springwood Road, Forest Acres, SC 29206-2113 Voice: (803) 790-9230 Fax: (803) 738-0218

© Copyright 1994 Development Technologies, Inc. All rights reserved. DeskMan, DeskMan/2 and VUEMan/2 are trademarks of Development Technologies, Inc. Workplace Shell is a trademark of International Business Machines Corporation. IBM and OS/2 are registered trademarks of International Business Machines Corporation.

Circle Reader Service Number 16

OS/2 Developer PRODUCTS & SERVICES GUIDE

CUSTOM SOFTWARE DEVELOPMENT



5408 HIGHWAY 290 WEST
AUSTIN, TEXAS 78735

OS/2
CROSS PLATFORM
CLIENT/SERVER
WINDOWS/NT
NOVELL NETWORK
NLN
APPLICATIONS & DRIVERS

512-892-6035

FAX: 512-892-1097

Circle Reader Service Number 18

YOUR *Single* SOURCE FOR OS/2 APPLICATIONS

UTILITIES	PRODUCTIVITY SOFTWARE
BOOKS & VIDEOS	GRAPHICS & CAD
OS/2 T-SHIRTS	COMMUNICATIONS
BACKUP SOFTWARE	NETWORKING
DEVELOPMENT TOOLS	WORD PROCESSING

ORDERS: 1-800-776-8284

FAX: 919-878-7479

INQUIRIES: 919-878-9700

3209 Gresham Lake Road, Suite 135 Raleigh, NC 27615



FULL COLOR
CATALOG
NOW AVAILABLE



Circle Reader Service Number 20

The easy to use 32 bit OS/2 PM IPF Language Editor

IPF Editor

- Designed for ease of use both by programmers and non-programmers
- Generate C and resource source files to automatically add context-sensitive help to applications
- Import wordprocessing files, including WordPerfect, to create on-line documents quickly and accurately
- Preview IPF file output without compiling
- Create all hypertext links automatically
- Improved creation of hypergraphic links
- Now available: Add voice and music to help and on-line documents with optional voice support module
- Create IPF tables from 1-2-3 and Excel spreadsheets
- Compatible with both Netware and LAN Server
- Includes complete on-line help

★ New - Modify on-line documents screen size, position

★ New - Multiple-language support

★ New - Footnote support

Perez Computing Services
4725 Monte Vista Place
Mount Vernon, WA 98273

*Price: \$150
Orders and
Information
(206)428-5025*

*on compuserve at
70410,2416*

IPF Editor and PCS are trademarks of Perez Computing Services.
All company and product names are trademarks of their respective owners.

Circle Reader Service Number 19

Attention Mathematica users:

Everything you need to know to get the most out of your math programming is in

THE MATHEMATICA JOURNAL

Call or fax today for information about subscriptions and special issues:

Phone orders: 415-905-2332

Fax orders: 415-905-2233

REACH 36,000+
ACTIVE, QUALIFIED
OS/2 DEVELOPERS
COST-EFFECTIVELY!

CONTACT: KRISTIN MORGAN

V/212.626-2275 F/212.869-0899

email: kmorgan@mfi.com

TO ADVERTISE IN THIS SECTION CALL: KRISTIN MORGAN @212.626.2498

PRODUCTS & SERVICES GUIDE

OS/2 Developer

Opt-Tech Sort/Merge

High Performance Sort/Merge/Select Utility

Use as a stand-alone utility or call as a Subroutine. Many features including unlimited file size, multiple keys, record selection, duplicate elimination, summing and more!

Call for free brochure Opt-Tech Data Processing
P.O. Box 678
Available for: Zephyr Cove, NV 89448
OS/2 or Unix \$249 Phone (702) 588-3737
DOS or Windows \$149 Fax (702) 588-7576

Circle Reader Service Number 21

FAX for OS/2

From the developers of **FaxWorks OS/2** and **PMfax**
Client/Server API and Printer Driver Toolkits
Support for LANs, up to 96 lines per CPU,
and all popular fax hardware

Keller Group Inc.

Voice: (612) 429-7273

Fax: (800) 329-3293

CIS: 72120,3404

Fax: (612) 653-1987

Faxworks is a trademark of SofNet, Inc.

PMfax is a trademark of Keller Group

Circle Reader Service Number 22

IRMWARE SERVICES

EDPT - Employee Development Planning and Tracking a low cost solution to fully automate Human Resource procedures (may be customized to meet your needs).

E/R DIAGRAMMER - An inexpensive entry into CASE technology that remains methodology independent without losing the precision of binary cardinalities.

IRMWare specializes in custom business solutions and software development.

FREE PRODUCT BROCHURE OR CONSULTING INFORMATION:

(602) 730-1510 or Fax (602) 413-0399 • P.O. Box 51093 • Phoenix, AZ 85076-1093

Circle Reader Service Number 23



REMOTE LAN. NETWORKING

RemoteVision extends LANs transparently over dial-up connections. RemoteVision provides access to a LAN from remote machines or remote LANs. Remote DOS, Windows, and OS/2 machines have a simulated network adapter.

Communications to the LAN is provided by the RemoteVision Server and async modems. Most LAN protocols and applications work, including NetBIOS, Named Pipes, APPC, 802.2, TCP/IP, and IPX. Call (415) 965-8607 and use option 2 for a free fax-back.

Circle Reader Service Number 24

TCP/2™

TCP/IP for OS/2®

Supports *all* released versions of OS/2
including 32bit API for OS/2 2.0 and above

Supports NDIS, CM, and ODI drivers
Network access from protected, real, and WINOS2 sessions

Full server operation for telnet, ftp, rsh and sendmail

IP gateway capability

NETBIOS, NFS ACCESS, SNMP TOOLS

DOMAIN NAME SERVER

DEVELOPERS KIT with socket library

VT102 & 3270 emulation for remote login

TCP™-OS/2 TRANSPORT MODULE.....\$ 200.00

A DAN LANCIANI PRODUCT

For further information contact: Essex Systems, Inc.
One Central Street, Middleton, MA 01949 • (508) 750-6200

TCP/2 is a trademark of DLD consulting. Ethernet is a trademark of Xerox Corporation, OS/2 is a registered trademark of International Business Machines Corporation. VT102 is registered trademark of Digital Equipment Corporation. TCP/2 is based in part on work done by the University of California Berkeley.

Circle Reader Service Number 25

TO ADVERTISE IN THIS SECTION CALL: KRISTIN MORGAN @212.626.2498



Object-Oriented Programming

*For the last year and a half, I have been working on a set of class libraries intended to provide a virtual operating system, network, GUI, and run-time support system for OS/2 applications. I have learned many things and hit many brick walls. Here, I share some juicy tidbits to aid your OS/2 development efforts. Although this article is ostensibly C++ oriented, it leans toward giving you insights into OS/2 that are necessary to develop a powerful class system. **By DEAN RODDEY***

An Example OS/2 C++ Class Library System



Dean Roddey

Any such class library project is a compromise between supporting OS/2's power and making it somewhat portable to other platforms. Although OS/2 catches much grief in the press, its problems are political, not technical. Sadly, a portable class system cannot make use of many Presentation Manager services because the Workplace Shell is so far ahead of the field at this time. Although my project covers kernel and GUI services, this article deals only with GUI issues. There are times, however, when I must discuss kernel services to fully cover the subject.

THE BASICS

The most basic issue for creating a class system over Presentation Manager is wrapping Presentation Manager windows into a class. If each window is an instantiation of a window object, there must be a way to link the window to the window object. There are two reasons for this. First, the window object must operate on the Presentation Manager window in response to calls to its methods. Second, window events coming to the window must be mapped to window class methods.

The first issue is simple; the window object can store away a handle to the

window. The other side of the equation is more difficult because window events arrive at the window without knowledge of the C++ language. The most common approach to this problem is to use the window's extra data to store a pointer to the window object. When a message comes in, the window procedure extracts the pointer to the window object and calls the correct methods for the event.

There are a number of things to watch out for. A window procedure cannot be a class method since all class methods must have an implied pointer to the object. But window procedures are called from Presentation Manager and must have the prescribed window procedure format. The answer is to create a protected friend method that acts as a gateway into the window class. A friend procedure does not have an object pointer parameter, but it can call protected class methods. Window messages come to the friend method, which extracts the window object from the window's extra data and calls the real window procedure method.

You should have only one main window procedure in the entire program so all messages will come through it. To achieve this, all classes that encapsulate existing Presentation



Manager windows should subclass the control using the window class's friend method. Custom classes can use the friend method as their window procedure when they register their window class. This seems like a lot of effort and somewhat backward, since the whole point of Presentation Manager is to allow each window class to provide its own layer of window procedure, allowing most messages to fall through to the layer below.

But C++ offers a much more powerful layering capability in its class system. Keep in mind that Presentation Manager's window procedure system is a grand compromise intended to offer, to a degree, inheritance mechanisms to procedural languages. In doing so, the window procedure system abandons strict type checking to achieve a pseudo run-time linkage system, similar to the C++ virtual method. C++ can provide all those layering capabilities and more while maintaining stricter type checking.

To provide the layering system that will work with Presentation Manager, the basic window class should provide a number of virtual methods that correspond to the window events it wants the window objects to respond to. Also, it should store, as a data member, the address of the default window procedure that will be called if the derived window class does not want to handle an event method.

For classes that encapsulate Presentation Manager windows, this would be the Presentation Manager control's window procedure. For custom classes, it should be the default Presentation Manager proc. Each event method should return a Boolean value that indicates whether or not it goes to the default proc. If the derived class handles the event, it returns `TRUE`. Then, the window procedure method will return

the correct "I handled it" value for that type of window event. If the derived class wants to let the event go to the default handler, it returns `FALSE`. It can either not override the event method (in which case the default version provided by the basic window class will return `FALSE`) or provide some processing and return `FALSE` to let the event go to the default handler.

When a virtual method is called, it is vectored to the highest level class that has overridden it. This class can handle the event, call its parent's version and provide more processing, or return `FALSE` to force the event to go to the default handler. This allows for the incremental improvement of functionality that C++ is so good at. But beware that we live in an imperfect world: sticky issues arise when you implement a cleanly layered system on top of a system that it cannot control totally.

STICKY ISSUES

There is a nasty side effect to this kind of system. Since the entire structure is based on the fact that all windows have a pointer to a window object, there is no easy way to support existing non-C++ code within this framework. If you create an non-C++ window as a child of one of your new objectified windows, it will almost certainly post or send messages to its parent or owner. This message will go to the central window procedure, which will attempt to extract a pointer to the window object and call methods therein. At this point, your application will crash. Certain schemes could solve this problem, but they involve too much overhead and complexity, unless you are desperate.

Another sticky issue involves `WM_CREATE` messages. This is not an issue for the classes that encapsulate Presentation Manager controls because you will probably just subclass them. Because



subclassing cannot happen until the window is created, you will never see the `WM_CREATE` message. But there is a problem when you are dealing with custom window classes. If you add a virtual `bCreate()` method to your basic window class, you will discover a

quirk of C++: when you call a constructor for a class that is high up the class hierarchy, the construction actually starts at the bottom. This is necessary, since each class level needs to call its parent constructor. Eventually, the call chain reaches the lowest level construc-

tor, which completes its function and starts back upward.

In my system, the actual call to `WinCreateWindow()` takes place at the lowest-level `WINDOW` class. So, when the `WM_CREATE` message arrives, the object is really a `WINDOW` object, and the virtual method will not reach the original calling class. This makes sense because that class has not initialized its members and would not be in a position to deal with the event.

But this is not a big issue. The reason for the `WM_CREATE` message is to initialize the window state. But in our objectified system, the constructor serves that purpose. So, have the window procedure method eat the `WM_CREATE` message.

PRESENTATION SPACES

The vast majority of Presentation Manager windows use cached presentations spaces, meaning that they only exist for the length of time of the `WM_PAINT` message. This makes sense; keeping 100K of memory or more (the amount needed for a persistent presentation space) for every cell in a spreadsheet doesn't make sense when it is needed infrequently. Other windows use more complex presentation spaces that are persistent and use different coordinate systems.

Here is a list of some potential problems with presentations spaces:

- Persistent and cached presentation spaces must be created and destroyed differently. Also, any invalidation of a `SYNCPAINT` window during `WM_PAINT` processing will cause a recursive call to the `WM_PAINT` code.
- Advanced applications often paint to a single window via multiple threads, each of which really needs its own presentation space.
- Two mutually exclusive color schemes (indexed or RGB) may be used.

Supercharge Your REXX!

GAMMATECHTM REXX SuperSet/2

Supercharge your REXX with the most complete set of REXX external functions available - GammaTech REXX SuperSet/2.

With over 300 functions from seven DLLs, the GammaTech REXX SuperSet/2 will initiate File and System operations, issue Network commands, execute Video I/O, manipulate Processes and Semaphores, regulate the MacroSpace and perform Math calculations.

Call us today at (405) 947-8080 and you'll see what REXX can do with some extra power!

- Perform EXECIO functions
- Manage LAN Server users by group or domain
- Perform LAN Server permission functions
- Manipulate LAN Server files, sessions and shares
- Interface with NetBIOS, TCP/IP and Communications Manager/2
- Open, close, query and create semaphores
- Start a thread
- Destroy a process or thread
- Copy, move or rename individual or mass files
- Dump variable pools or MacroSpace to a file
- Load, drop or reorganize functions in MacroSpace
- Perform logarithmic and trigonometric calculations
- Query hardware components
- List current system environment variables
- Write character and attribute strings
- Obtain and set cursor type and screen mode



**SoftTouch Systems, Inc.
Workstation Division**

1300 S Meridian, Suite 600, Oklahoma City, OK 73108
(405) 947-8080 • Fax (405) 632-6537

©GammaTech, Inc. 1991-94 All Rights Reserved

Circle Reader Service Number 29

- Some GUI resources are tied to the presentation space on which they were created and become useless when the presentation space is destroyed.
- Each layer of the class system cannot assume the state of the presentation space that they are given to paint on. Nor can they modify it, unless that is the purpose of the method that is called.

We dealt with the first two problems by creating a **PAINTBRUSH** class. This class's constructor accepts a window object that it should operate on. It extracts the window handle and creates a presentation space from it. This **PAINTBRUSH** object then can be used to paint on the window. The **PAINTBRUSH** class provides methods for all desired graphics primitives. At first, we stored the presentation space in the window, but the recursive nature of painting and the occasional need to have multiple threads paint to the same window required a change to the architecture.

Even so, this new system is a compromise: now we have two objects that are linked via a common system resource. If the window object is destroyed, any further access to the **PAINTBRUSH** object will cause errors. There is a protected constructor available to the **WINDOW** class to handle the special case of **WM_PAINT** messages. And there is a third constructor that creates a persistent presentation space and accepts extra parms for page units. A flag in the **PAINTBRUSH** object indicates the type of presentation space, so the destructor can clean it up appropriately.

In my opinion, the second problem is insurmountable if you desire to create a consistent, plug-n-play environment. Therefore, we avoided the problem by using only RGB mode. This may be unacceptable to many because of memory usage issues, but the Year of the True Color Card is not far away,

and we are looking forward, not backward.

We created an **RGBCLR** class that handles all the niceties of RGB color values. We provided a global set of named **RGBCLR** objects that correspond to the usual 16 colors, plus some special ones. We pro-

vide a **CLRPAL** class for those applications that must use indexes. When converting **RGBCLR** objects to a form passable to GPI APIs, you can reduce overhead greatly by having the color component members in the same order as the Presentation Manager RGB values.



Smart-Lock

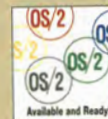
- a comprehensive security and configuration concept
- for host-connected OS/2 workstations
- based on Smart-Cards
- featuring the Chip-Controlled OS/2 Desktop



Smart-Lock is a security concept that controls the access to OS/2 systems by employing Smart-Card technology and a SOM-based restricted workplace shell.

Smart-Lock configures the OS/2 desktop according to the information on the E2PROM chip. Features include user specific boot protection and diskette access, single logon to LAN and host as well as the possibility to encrypt data and faxes.

Smart-Lock is CID enabled and allows easy installation and maintenance in large networks.



Distribution, installation and support in the U.S.A. and Canada through:



International, Inc.

621 NW 53rd Street, Suite 240
Boca Raton, FL 33487 USA
1-800-4SE-INTL or (407) 241-3428
Fax: (407) 278-3919

Circle Reader Service Number 30



Then, you can provide an implicit conversion method that will reference the first member and dereference it as a long value. This allows RGBCLR objects to be passed directly to GPI APIs with no intervening conversion to a temp variable.

The third problem is a problem primarily for logical font and bitmap IDs. Since fonts pose particular problems, I will cover this third problem shortly.

We dealt with the fourth problem by creating a GRAPHICATTR class, whose constructor accepts a PAINTBRUSH object and an attribute bundle identifier. Presentation Manager supports a number of attribute sets that can be saved and restored as a group. The GRAPHICATTR class saves a particular bundle and restores it when it is destroyed. This greatly simplifies the logic of the medium-level graphics output methods that perform lower-level operations on behalf of the caller. They simply create GRAPHICATTR objects to store the attributes they will modify. Any exit from the method destroys the object, restoring the caller's attributes. Of course, this saving and restoring of attributes should be minimized because it has performance costs. Another possibility is to add the Presentation Manager attribute push and pop mechanism to the PAINTBRUSH class, but it would not have the same automatic cleanup mechanism as a separate object.

FONTs

Fonts are sticky; they usually are installed by the user, rather than built into the operating system. They have numerous parameters and a number of formats. Sometimes bugs are in the font files themselves. If you look at any commercial package (object oriented or otherwise) you can easily recognize that fonts can be complicated. Some packages don't support fonts at all, and others pro-

```
// This is the skeleton of the basic WINDOW class.
//

class WINDOW
{
public :
    // Constructors/destructors
    WINDOW(...);
    //
    // Virtual 'event' methods handle incoming events from
    // PM. These defaults all return FALSE, to let the
    // event go to the default handler.
    //
    virtual BOOL bDestroy() {return FALSE;}
    virtual BOOL bPaint(AREA areaInvalid) {return FALSE;}
    virtual BOOL bSizeChange(AREA areaNew
        , AREA areaOld) {return FALSE;}

    //
    // Non-virtual methods handle operations on windows from
    // client code.
    //
    VOID Hide();
    VOID Show();
    VOID Size(AREA areaNew);

protected :
    // A friend that is the actual PM window proc
    friend MRESULT EXPENTRY _mresPreprocessor(HWND,ULONG,MPARAM,MPARAM);
    // The method that is called by mresPreprocessor
    MRESULT _mresMsgHandler(HWND, ULONG, MPARAM, MPARAM);
private :
    HWND    __hwnd;
    PFNWP   __DefWndProc;
    ULONG    __ulId;
};
//

//
// The friend function that is the actual window procedure. This
// guy is a friend but cannot access private class members or
// data.
//
MRESULT _mresPreprocessor( HWND hwnd, ULONG ulMsg, MPARAM mp1, MPARAM mp2)
{
    //
    // Eat the WM_CREATE message, but store away the passed object
    // pointer.
    if (ulMsg == WM_CREATE)
    {
        WinSetWindowPtr(hwnd, QWL_USER, mp1);
        return 0;
    }
}
```

Figure 1. Skeleton of the basic Window class (continued on page 57)



```
//  
// Extract the object pointer and call the message  
// handler method.  
//  
WINDOW* pwndThis = WinQueryWindowPtr(hwnd, QWL_USER);  
return pwndThis->mresMsgHandler(hwnd, ulMsg, mp1, mp2);  
};  
//  
// This class method is called by the actual window proc. This  
// guy has accessed to window object data and methods.  
//  
MRESULT WINDOW::_mresMsgHandler(HWND hwnd  
    , ULONG ulMsg  
    , MPARAM mp1  
    , MPARAM mp2)  
{  
    if (ulMsg == WM_PAINT)  
    {  
        RECTL rectlInvalid;  
        HPS hPS;  
        // Prep the PS and get the invalid rectl  
        hPS = WinBeginPaint(_hwndThis, 0, &rectlInvalid);  
        //  
        // We call a special PAINTBRUSH constructor that is for  
        // the special case of WM_PAINT. We convert the invalid  
        // rectangle to an area to pass along.  
        //  
        AREA areaInvalid(rectlInvalid, eNONINCLUSIVE);  
        PAINTBRUSH ptbTarget(hPS);  
        //  
        // If the class handles it, return the correct code.  
        // Otherwise, let it go to the default handler.  
        //  
        if (bPaint(ptbTarget, areaInvalid))  
            return 0;  
        //  
        // Note: No WinEndPaint()!! The PAINTBRUSH's destructor  
        // will handle this. It will be destroyed on the  
        // exit from this block.  
        //  
    }  
    else if (ulMsg == WM_SIZE)  
    {  
        // Convert the PM info into areas  
        AREA areaNew(0, 0, SHORT1FROMMP(mp1), SHORT2FROMMP(mp1)); AREA areaOld(0,  
0, SHORT1FROMMP(mp2), SHORT2FROMMP(mp2));  
        //  
    }  
}
```

Figure 1. Skeleton of the basic Window class (continued on page 60)

vide very limited support. On this issue, you may have to bite the bullet. If your applications require heavy font usage, and no class system is available that can provide that support on the desired platforms, you may have to back off some platforms or do a lot of per-platform coding.

OUR APPROACH

We abstract fonts very heavily due to their inherent cross-platform complexity and because there are many font problem areas just in Presentation Manager. We created a FONTINFO class that is just a front end to the FONTMETRICS structure of Presentation Manager. The application can request a linked list of FONTINFO objects that match a face name and a somewhat limited set of attributes. We tried to limit the attribute set to a reasonable subset that would be useful yet portable, but there will probably be problems in the first port. Some attributes may just have to be ignored on some platforms.

Also, only a selected set of FONTMETRICS fields is accessible so that extremely Presentation Manager-centric font manipulations are not spread around the application.

To use a font, you must create a logical font ID. As discussed previously, when a presentation space is destroyed, the current font selection is lost. Also, in subsequent platform ports, you cannot always foresee the associations between window state and fonts.

For these reasons, and because there are only 254 font IDs available per process in Presentation Manager, we decided to keep strict control over fonts, to the extent that we do not have a font class. Instead, the application finds FONTINFO objects that represent its desired fonts and asks a PAINTBRUSH object to create a font of that type. When it wants to use the font, the application passes in a font

name that it will use later.

So, the `PAINTBRUSH` keeps a list of font IDs associated with it. This puts the `PAINTBRUSH` in a position to manipulate font IDs to match its current state as needed. It also allows it to destroy any associated font IDs when the presentation

space is destroyed to avoid a common error of stranding font IDs and running out of them later.

DIALOG WINDOWS

Dialog windows pose more problems. There are some basic approaches: You can call `WinLoadDlg()`,

which will load up the dialog and create all the controls. Then, you can create after-the-fact window objects for all the controls. But we wanted to allow each dialog-window-derived class to control the creation of the control windows so that more specialized versions could be created and initialized in a way that is appropriate for that dialog. We could have passed the window handle and Presentation Manager window class of each of the controls to the derived class, but that would expose Presentation Manager-specific information to the outside world.

We solved this problem by interpreting the dialog template manually, allowing the basic dialog class (or a derived class) to handle the creation of the controls. Reading a dialog template is not as difficult as it might seem, except that the documentation is extremely insufficient. Again, do not create a virtual method to allow derived classes to handle control creation because the creation happens in the constructor. We used a callback function passed to the constructor. The Presentation Manager class type is converted to the equivalent class name, and any control size and position information is passed along. If the callback returns a null pointer, the dialog window creates the default type of window object.

By the way, another good use for this knowledge is for building dialog templates in memory on the fly and letting the dialog processing code handle the grunt work of user interaction.

The devil is in the details, but these are the basic steps to handle your own dialog creation. First, use `DosGetResource()` to load the dialog resource. This gets you a read-only buffer with the template definition. Load the address into a `PDLGTEMPLATE` variable to start the parsing. The last field of the `DLGTEM-`

Rhetorex Voice Processing boards make CTI a reality.



If you're asking "what's CTI," you're missing one of the hottest new technologies going.

Computer Telephony Integration links PC-based computer applications to the telephone network, providing voice/fax mail, interactive voice response, and other telephony applications.

Interested? Maybe you're already developing a CTI application. Then it's time to discover Rhetorex.[™]

The fact is, when it comes to multi-port voice processing components, no one offers more value than Rhetorex.

With Rhetorex voice platforms, you'll get state-of-the-art DSP-based technology, designed from

the ground up for reliability. Sophisticated DSP algorithms give you unparalleled performance. Best of all, enhancements to algorithms are made via software upgrade.

And Rhetorex products help you develop applications quickly. All components include device drivers, a straightforward API, and sample programs. Friendly, free technical support is available at any stage of development.

For the best value in CTI technology—from our 2 and 4 port voice processing boards and fax boards, to our 24-port platform—then give Rhetorex a call. (408) 370-0881. And start making CTI a reality.



RHETOREX

Rhetorex, Inc.
200 E. Hacienda Avenue
Campbell, CA 95008-6617
Tel. (408) 370-0881
Fax (408) 370-1171

All trademarks identified by the [™] symbol are trademarks of Rhetorex, Inc.
All other trademarks belong to their respective owners. © 1993 Rhetorex, Inc.

Circle Reader Service Number 31

PLATE structure is an array of DLGITEM records, but the array has only one entry since the actual number of dialog items following the template is variable.

Use a PDLGITEM variable to work your way through the dialog item structures (pItem = &pDlg->adlgti[0]). The 0th dialog item represents the dialog window itself. Each dialog item structure contains the information you need to create that item and the number of child windows it has. The child windows follow it in the list, so the arrangement is "depth first." In our system, we do not support nested dialog controls, so we worry only about the number of children of the dialog itself.

Next, create the dialog window frame using the information in the first dialog item structure. You will need to check the styles to find out whether the size and position information is relative to the screen, parent window, or mouse and then add the dialog position to the origin of the screen or window or to the current mouse position. The control data for the dialog is a ULONG with the FCF_XXXX flags in it. You need this to create the frame window with the correct frame controls.

For the modal processing to work correctly, you must calculate the correct owner of the dialog, which will not always be the owner window that the user passed. So, starting with the owner window that the user indicated, use WinQueryWindow() to work up the ownership chain until you reach a window that is a direct child of the desktop window or a null window handle. Pass this window as the owner of the frame window. If you do not do this, often you will find that the supposedly modal dialog does not stop you from interacting with other application windows.

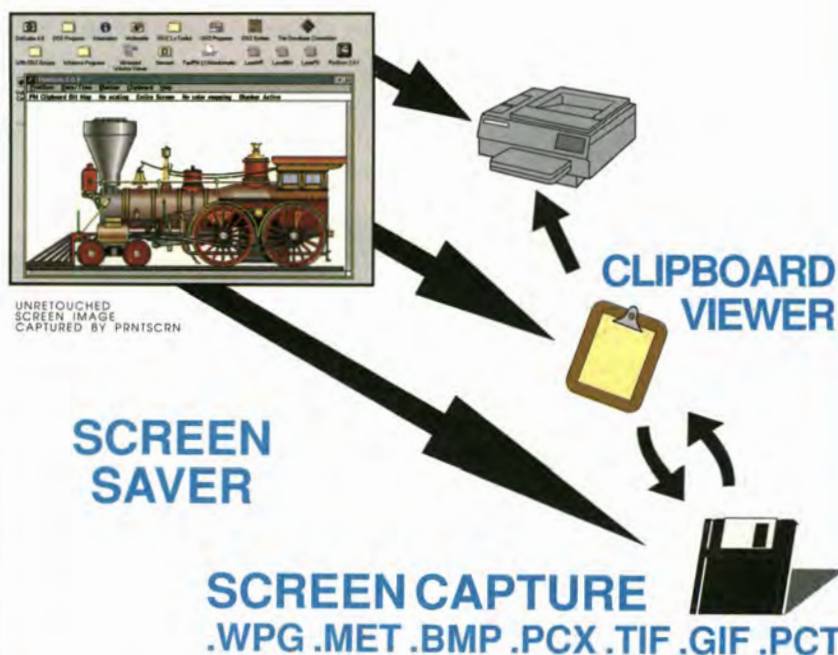
Now, call WinCreateWindow() to create the frame, passing a FRAMEC- DATA structure as the control data.

At this point, just create it so it is invisible and has no size. You don't know the size at this point because the size information in the dialog item is actually the size of the "client area" of the frame (although there is no real client window in a dialog).

After frame creation, use the dialog item size and position information in a call to WinCalcFrameRect() to calculate a frame that will contain a client area of that size. Before you size and position the frame, adjust the values to ensure that it does not hang off the screen. Then,



SINGLE KEY-STROKE SCREEN PRINTING



- Quality Color or Black & White copies of color Desktop Images.
- Copy any portion of any image on the Desktop.
- Screen Saver prevents monitor burn-in.
- Date & Time Display for Time-Stamping images.
- LAN installable. Complete support for LAN attached printers.
- Economical! 4 Utilities for 1 Price! Entity licensing available.
- The most stable and reliable product available.
- Includes both OS/2 2.1/2.0 (32-bit) and OS/2 1.3 (16-bit) versions.

THE LEADER: **PrntScrnm**TM Screen Image Utility for OS/2



MITNOR Software
Phone: (918) 357-1628
Fax: (918) 357-2869

SCREEN IMAGES IN OS/2 DEVELOPER ARTICLES ARE CAPTURED USING PRNTSCRNM, COURTESY OF MITNOR SOFTWARE

Circle Reader Service Number 32

you can size and position the frame (but not show if you want to process it modally). To make the client area display the correct color, use `WinQuerySysColor()` to get the `SYSCLR_DIALOGBACKGROUND` color. Call `WinSetPresParam` to set the frame's `PP_BACKGROUND` color.

Now, you can loop through the remaining dialog items (which represent dialog controls) and create them as owned children of the frame. One oddity is that you must start at the last control and work backward to make the Presentation Manager dialog window's tab key code work correctly. By going in reverse, the first control in each group (which has the `WS_GROUP` style) will be the last window created in the group.

If you intend to support nested controls, you should do this via a recursive call. Note that, since dialogs have no client window, you actually create the controls relative to the frame. The position information in each dialog item does not take this into account, so use the `WM_QUERYBORDER` message to get the border size and add it to the origin of each control.

All size and position information in dialog templates is in dialog units, not pels. Use the `WinMapDlgPoints()` function to map the dialog units to pels before using them. And last, the dialog template structure contains the index of the control that should get the initial focus. If it is `0xFFFF`, use `WinEnumDlgItem()` with the `EDI_FIRSTTABITEM` to find the first item with the `WS_TABSTOP` style and give it the focus.

Now, you can either use the newly created window like a regular window (which is useful for creating your program's main windows via the Dialog Editor) or provide a method in the dialog class that calls `WinProcessDlg()` to process the dialog modally. Initially set the default procedure to the default window procedure. If the dialog is

```
// If he handled it, return correct code, else let it go
// to default handler.
//
// if (bSize(areaNew, areaOld))
//     return 0;
//
// else if (ulMsg == xxxxxx)
// {
//     ... Whatever other messages you want to handle
// }
return _DefWndProc(hwnd, ulMsg, mp1, mp2);
}
```

Figure 1. Skeleton of the basic `Window` class (continued from page 57)

later processed modally, switch it to the default dialog proc to avoid problems.

POINTS AND RECTANGLES

One of the best things you can do for yourself in a GUI class system is to create powerful and flexible classes for dealing with points and rectangles. There are three reasons for this.

First, two types of rectangles are in Presentation Manager, inclusive and noninclusive. In Presentation Manager applications, a constant problem is keeping up with which kind of rectangle is required and which kind you have.

Second, any new platform you port to may have totally different concepts for rectangles.

Third, you waste a vast amount of time in the average, large Presentation Manager application doing grunt work calculations and manipulations on points and rectangles. Such code is inherently Presentation Manager-centric and ripe for maintenance errors.

We eliminated this problem by creating a `POINT` class to handle points and an `AREA` to deal with rectangles. The `AREA` class represents a `POINT` origin and a width and height value that is always inclusive. The outside world deals exclusively with `POINTS` and `AREAS`,

and the lowest-level class methods (that actually call Presentation Manager) handle converting in and out as needed.

The `AREA` class is particularly powerful; it inflates and deflates itself, calculates its center, calculates horizontal and vertical percents of itself, centers other areas within itself, offsets its origin, and so on. The more standard operations you incorporate into the `AREA` class the better. It should have methods and constructors that build `AREAS` from `RECTs`, a `POINTL` and width and height, two `POINTLs` that represent opposite corners, `SWP` structure, and so on. It should also have methods to create these Presentation Manager structures from an `AREA`, which greatly simplifies the internal code that has to do the conversions.

The use of `POINT` objects can be eased greatly by storing the members in the same order as a `POINTL` and having an implicit conversion operator that references the address of the `x` member. This allows `POINT` objects to be passed directly to GPI calls with no intervening conversion.

PRESENTATION MANAGER TYPES AND BIT FLAGS

One way to increase the type safety and portability of your class

system (without introducing overhead) is to redefine the Presentation Manager types as C++ enumerated values. C++ allows enumerated values to be given explicit values, so you can create, for instance, an enumerated value called `eLINESYLES` for the `LINE_TYPE_XXXX` Presentation Manager values.

If your line type setting method accepts an `eLINESYLES` parm, it gets much stricter type checking than the generic bit flags that Presentation Manager uses. But, internally, the line style setting method can pass the enumerated value on to the GPI API, and the compiler will convert the enumerated value automatically to a long value. A side effect of this trick is that you keep Presentation Manager-centric constants and types out of the source code, easing future ports. It also allows the class designer to exclude certain flags because those flags may limit portability.

THE BIG PICTURE

If you decide to write your own Presentation Manager class system, be aware that you likely will be very frustrated. C++ represents such a great step in power over the procedural systems (upon which most class systems must be implemented) that a totally consistent class system might be impossible.

We started the classes by mainly implementing basically runtime-oriented services, such as strings and containers. Since these services depend only on memory allocation, they are free to follow almost any consistent class mechanisms you implement. The kernel services that followed were also quite easy because most kernel services are passive and do only what you tell them. Threads were not too difficult.

Then came the GUI, whose complexity, event-driven nature,

and troublesome resource interrelationships often stymied my plans. This is irksome even now because I feel that consistency is next to godliness in class system development. Developer spin-up time and reactionary fire-drill programming are very expensive, and inconsistency greatly increases spin up and decreases reliability. A class system in total control can provide many more safety nets and self examinations, quickening development and relieving much of the testing burden for the developer and putting it into the code itself.

The object-oriented frameworks and (eventually) operating systems to come from IBM will reduce these problems greatly by providing an object-oriented platform to build on. Of course, the irony is that they also will make

most of the effort that I have discussed here redundant.

If you can't wait, be sure to take a good look at the operating methods of your future object-oriented program's framework. Then, arrange your class system accordingly. Hopefully, you will be able to just pull out your lower-level classes and replace them with new ones, leaving your higher-level (application-specific) classes mostly intact. It may never happen, but you have to try!

Dean Roddey is senior software engineer at Quantitative Medicine of Annapolis, Md. He has a relationship with his computer that is probably illegal in some states, but he is seeking help. He is currently working to move QMI's OS/2-based Clinical Information System to an object-oriented platform.



GET FAST ANSWERS TO YOUR DEVELOPMENT QUESTIONS

You need to do two things:

- Go on-line with CompuServe®.
- Use Golden CommPass to do it.

CompuServe hosts OS/2 developer and user forums. IBM developers have responses to anything that's got you stuck. Other OS/2 programmers and enthusiasts are there, too, comparing notes and giving feedback. It's definitely the place to be.

Time is money when you connect to CompuServe, and Golden CommPass saves you both. It lets you compose your questions off-line, send them in a

flash and disconnect. It gets your replies while you're busy programming.

Golden CommPass keeps your development schedule on course. The fact is, the sooner you start using our program, the sooner they'll be talking about yours.



Golden CommPass®
OS/2® NAVIGATION SOFTWARE

Creative Systems Programming Corporation
(609) 234-1500 • Fax: (609) 234-1920
Internet: 71511.151@CompuServe.com



NEW FIBER OPTIC Pen Computing FOR THE DESKTOP

Under OS/2® 2.1, this new, fast input device allows you to take advantage of all the new drag and drop icons for fast target selection. Run everything the mouse does and more. Frustrated because you can't sign your name with a mouse? Add IBM's Pen for OS/2 software to create gesturing, sign your name and use handwriting recognition. Finally, click and go on your desktop monitor is a reality.

WHAT COULD BE EASIER?

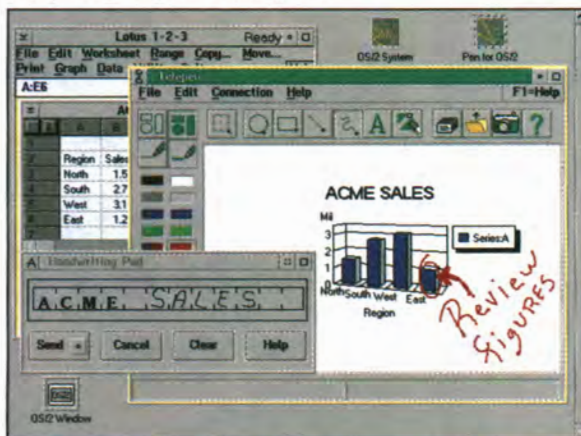
How much easier can pointing your finger be? Just hold the PEN in your hand and tap the tip at any CRT computer screen - your computer executes the command and you're off. A mouse simply cannot keep up!

GET YOUR DESK BACK.

Increase speed and accuracy and save desk space at the same time! One reason PENS are so easy to use is that they don't clutter up your desk with pads and hardware. You don't need any desk space for the PEN. The PEN stores neatly on the side of your monitor in a convenient holder.



WARP SPEED™ has developed a NEW fiber-optic PEN for any computer running OS/2.



Handwriting recognition, gestures, notes etc. are all made simple with the addition of IBM's Pen for OS/2 software.

CURSOR, CURSOR, WHERE'S THE CURSOR?

This game just became obsolete. You no longer need to go on a mouse hunt to find your cursor. Just ignore the cursor, touch the PEN to the target... the cursor just found you, instantly. This happens so incredibly fast you'll think the cursor is invisibly attached to the PEN.

JUST PLUG AND GO.

The WARP SPEED PEN for OS/2 comes complete with an adapter card (ISA or MCA), a high quality Stainless Steel PEN and PEN driver and calibration software for OS/2. Gesturing and handwriting recognition are available on IBM's Pen for OS/2 software.

RISK FREE INTRODUCTORY OFFER.

Make the jump to a WARP SPEED PEN for OS/2, try it for 30 days. If you don't agree that the PEN is far superior to the mouse, simply return it to us in its original box, and we will refund your money, no questions asked.



Save
\$90.00

Call TOLL FREE (800) 874-4315 or (505) 258-5713 to order your WARP SPEED PEN for OS/2 and we will include IBM's Pen for OS/2 software for the low price of \$199.99. You save \$90.00 off the retail price of \$289.98 for both products. This is a limited time offer. Use your credit card or mail a check to Warp Speed, 1086 Mechem Drive, Ruidoso, NM 88345.

WARP SPEED is a trademark of Warp Speed Light Pens Inc. OS/2 and IBM are Trademarks of IBM Corporation.

WARP SPEED™



The IBM Pen for OS/2 product enables applications to cross the human-centric barrier. This article explores an example of how a pen-centric application should be designed and how simple it can be to develop using Pen for OS/2. By SARKA J. MARTINEZ

A Natural User Interface: Writing a Pen-Centric Application

Pen technology has introduced a whole new set of possibilities in the world of application software. Software can now be written in pen-centric fashion. Pen-centric applications are those that have been written expressly with pen entry and manipulation in mind. A pen-centric application can perform specific preprogrammed tasks as a result of what a user does with the pen, be it handwriting or gesture entry.

Application programming takes on an exciting aspect when coupled with OS/2's pen extensions. One such pen-centric application is called Contact/2. Contact/2 is a demo application I wrote in five weeks while working in IBM's Mobile, Pen, and Voice (MVP) software development team. A unique approach was taken in developing Contact/2, the core ideas being the pursuit of an interface using a real-world paradigm, the value of function, and of course an easy-to-use pen-centric approach. This application is an example of how you can add pizzazz to your applications and improve their ease of use by the mere fact that they follow a real-world model.

CONTACT/2: A PEN-CENTRIC DEMO

Basically, Contact/2 is an address book. Its intention is to communicate with the people and business contacts stored within

the book. This application achieves its entry into the real-world paradigm by being represented as a true-life book, complete with a cover, book tabs, business cards, and an index page. There's even the capability to attach jotted down memos via sticky pads to any entry in the book.

Contact/2 can give the user a simple way to communicate with other people via fax, e-mail, telephone, alphanumeric paging, or LAN messaging. It includes some multimedia capabilities; it allows the playback of prerecorded messages attached to the cover, the index page, and the book's entries (so that the people in your book can tell you a little about themselves). Communications functions were tied in easily by the use of OS/2's dynamic data exchange (DDE). By using DDE and off-the-shelf applications, we saved a lot of time and money by not having to develop new code or compete with existing communication software.

THE DESIGN APPROACH

Our approach in designing this particular application was different. The emphasis was on visuals—something often saved for last. The coding was merely an enabler for the end result. Painstaking care was taken up front to create the look and feel of an actual book, including the artwork for its dramatic cover.



Sarka J. Martinez

LOOK AND FEEL FIRST

First we designed the book's overall visual appearance and user interface using PowerPoint 3.0. Next, the graphics for the cover (welcome page) of the book were created using CorelDraw 3.0 and Adobe Photoshop 2.5. The images were saved as bitmaps and are displayed on the welcome page when its dialog box is loaded.

Notebook Control—Made to Order. After the user interface design and visuals were created, work began on implementing the interface using the Notebook control. If there has ever been a question as to when to use the Notebook control, this sort of application is what it was intended for! There is no better vehicle in representing this type of information to a user.

After we had finished the user interface framework, we created a database to store address book entry information. Along with storing the usual information, we added the ability to store a bitmap for each of the entries. We created generic bitmaps, using OS/2's Icon Editor for male and female entries, and a bitmap to indicate a business entry. We provided a way for users to add their own bitmaps.

Next, the pen portion was enabled by using several Pen for OS/2 APIs and messages. Last, DDE was folded into the code, enabling Contact/2 to communicate with various off-the-shelf "helper" applications such as IBM's Multimedia Extensions, cc:Mail, and other communicating applications. From design to coding, everything was done on OS/2 2.1.

THE GESTURES

Pen for OS/2 APIs follow the same format as other OS/2 2.1 APIs, making it easy to learn. Anyone creating software can be a quick study in getting pen capability into their applications. Let's take a look at how simple it is.

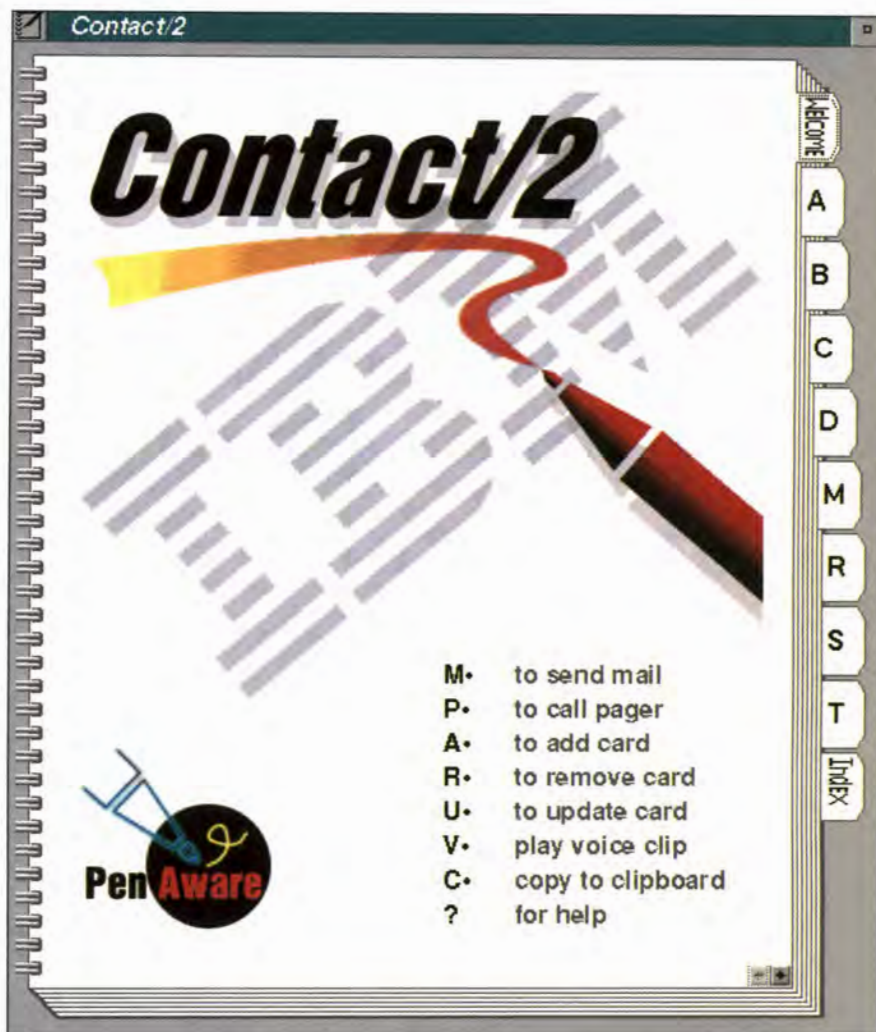


Figure 1. The cover of the Contact/2 address book

Pen plays a major role in Contact/2's book paradigm in both navigation and the initiation of its communication features. Following is a list of Contact/2 premapped gestures achieved by using the Pen for OS/2 messaging framework. The gestures defined by Pen for OS/2 retain their predefined function unless overridden by another application's mapping. There are 58 gestures available for mapping:

->	Forward a page
<-	Backward a page
R.	Remove entry
A.	Add entry
U.	Update entry
C.	Copy
?	Help

D.	Print (output)
P.	Send Pager Message
D.	Dial a number
V.	Play a voice capsule
M.	Mail a note
F.	send a fax
L.	send a LAN message.

Letter gestures involve a tap of the pen as the final stroke.

How does Pen for OS/2 determine if a user is entering a gesture rather than a letter (handwriting recognition)? It all comes down to pauses. As long as a user does not pause the pen on the tablet before or during the execution of drawing a gesture, the system will treat it as a gesture. If a pause is present, the system will attempt to recognize it as a letter.

RECOGNITION

Gesture and handwriting recognition can be monitored and acted upon by processing the `WM_RECO` message. An application receives a `WM_RECO` message whenever a recognition event occurs. The *Contact/2* application redefines several gestures. Thus, the application performs special processing of the `WM_RECO` event to override the system's default processing of the gesture.

In our application, several window procedures are defined; each time a recognition event occurs on one of these windows, I repost the parameters that enter into the `WM_RECO` event into a mes-

sage called `WM_MY_RECO`, which is defined in the topmost client window (the *Notebook* control). By doing this, we need to define only one procedure to handle all the gesture processing information.

When you process a `WM_RECO` message, the second message parameter contains the `RECODATA` structure. A virtual ID can be accessed through this structure. The virtual ID indicates whether the gesture was a letter gesture or another specific symbol gesture. If it is determined that the gesture was a letter gesture, the character code stored in the `RECODATA` structure can be accessed. This will let you determine exactly which let-

ter gesture was recognized.

Finally, a value of `TRUE` must be returned from the `WM_RECO` event to override the system's default execution of the gesture. Notice that the gesture's hot-spot coordinates can be mapped easily to the corresponding window handle that resides at that coordinate. This can be achieved by using the `WinMapWindowPoints` API, which is detailed in Figure 3.

Handwriting recognition can also be achieved easily with the use of the *Handwriting* control. No special message capture or message sending needs to be made. Handwritten words are recognized automatically as long as a value of `FALSE` is returned from the `WM_CONTROL` event. This can be done by returning the result from `WinDefWindowProc` in the `WM_CONTROL`'s default case statement. This causes the system to take default processing of handwriting recognition events. The code segment in Figure 3 shows how to process recognition messages for gestures.

THE SKETCH CONTROL

A user can access the sticky pad by tapping on the paper clip button. This interface is intended to let a user jot down quick notes about an entry. *Contact/2*'s sticky pads were created using the sketch control provided by *Pen for OS/2*. Finally, a way to jot down thoughts through sketching! This is shown in Figure 4.

The sketch control was included into a dialog box within a `.DLG` file. The dialog procedure needs to process `SKN_STROKE_ADD` messages to save the strokes that a user may draw in to the control. Then, these strokes can be saved into a file and later be retrieved for display purposes. This is shown in Figure 6. These saved strokes could also later be recognized and transformed into text (delayed recognition).



Figure 2. Entries are represented as business cards


```

MRESULT APIENTRY NoteBookClientProc(HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2 )
{
    ...
    case WM_RECO:
        // post the message back to this window, all windows will post this message back to this window...
        WinPostMsg(hwndClient, WM_MY_RECO, (MPARAM)hwnd, mp2);
        // need to return true...
        return (MRESULT)TRUE;

        // look at all gestures done on top of Contact/2
    case WM_MY_RECO:
        ProcessReco(mp1, mp2);
        break;
    ...
}

MRESULT EXPENTRY BusinessCardDlgProc( HWND hwnd, ULONG message, MPARAM mp1, MPARAM mp2 )
{
    ...
    case WM_RECO:
        // post the message back to the client window, so that all gesture work stems from there.
        WinPostMsg(hwndClient, WM_MY_RECO, (MPARAM)hwnd, mp2);
        // need to return true...
        return (MRESULT)TRUE;
    ...
}

VOID ProcessReco(MPARAM mp1, MPARAM mp2)
{
    RECODATA *recodata;
    ...
    // someone drew a letter gesture
    if (recodata->virtual_id == VE_LETTERGESTURE) {
        // user wants to play back a greeting
        if (recodata->char_code == 'V') {
            ...
            index = (ULONG) WinQueryWindowULong(hwndDlg, 0);
            // find out which card the gesture was done on
            WinMapWindowPoints(HWND_DESKTOP, hwndDlg,
                               &(recodata->ptlHotSpot), (LONG)1);
            WinQueryWindowRect(hwndDlg, &rect);
            // playback the entries voice message, since you know what card the gesture was done on
            ...
        }
        // user wants to send a Pager transmission
        else if (recodata->char_code == 'P') {
            ...
        }
        // user wants to send Mail
        else if (recodata->char_code == 'M') {
            ...
        }
        // user wants to Add an entry to the book
        else if (recodata->char_code == 'A') {
            ...
        }
        // other gestures to take action on
        ...
    }
}

```

Figure 3. Code enabling gesture recognition (continued on page 67)



```

    }
    // bring up help
    else if (recodata->virtual_id == VE_HELP) {
        ...
    }
    // turn back a page
    else if (recodata->virtual_id == VE_SCROLLLEFT) { ...
        // turn to the index page
        if (hwndDlg == hwndWelcome)
            WinSendMsg(hwndNB, BKM_TURNTOPAGE, MPFROMLONG(IndexCardId), 0L);
        else if (hwndDlg == hwndIndex) {
            PageID = (ULONG)WinSendMsg(hwndNB, BKM_QUERYPAGEID, MPFROMLONG(IndexCardId),
                MPFROM2SHORT(BKA_PREV, BKA_MAJOR));
            WinSendMsg(hwndNB, BKM_TURNTOPAGE, MPFROMLONG(PageID), 0L);
        }
        ...
    }
    // turn forward a page
    else if (recodata->virtual_id == VE_SCROLLRIGHT) {
        ...
    }
}

```

Figure 3. Code enabling gesture recognition (continued from page 66)



Figure 4. The sketch pad interface

Basically, the SKM_STROKE_ADD message stores all the strokes in a stroke buffer that can later be retrieved using the SKM_QUERY_CTL_STROKE_COUNT, SKM_QUERY_STROKE_LENGTH, and SKM_QUERY_STROKE_DATA messages. These messages can, in turn, allow you to store the information to a file. You can put stroke information back into a buffer to allow sketch replay. This can be done by inserting the stroke data into the SKETCH-STROKEDEF data structure. You can redisplay the stroke information stored in this data structure by sending the SKM_ADD_STROKE message to the sketch control. This is shown in Figure 7. Sending the message SKM_DELETE_ALL_STROKES to the sketch control allows the clearing out of the control's sketched contents. By sending the SKM_UNDO_LAST_STROKE message to the sketch control, a user is able to undo a stroke within a sketch, one stroke after another, until nothing is left. You can modify the pen's ink color sending the sketch control the SKM_SET_CTL_INK_COLOR


```

MRESULT EXPENTRY SketchDlgProc( HWND hwnd, ULONG message, MPARAM mp1, MPARAM mp2 )
{
    ...

    switch (message)
    {
        case WM_INITDLG:
            ...

            WinSendMsg(WinWindowFromID(hwnd, ID_SKETCH), SKM_SET_CTL_INK_COLOR, MPFROMSHORT(CLR_BLUE), 0L);
            ReadAndDisplaySketch(hwnd, CardInfo->RecId);
            return (MRESULT)TRUE;

        case WM_CONTROL:
            switch (HIUSHORT(mp1))
            {
                // add the stroke to the control's 'data base'
                case SKN_STROKE_ADD:
                    return 0L;

                // clear entire control
                case SKN_CONTROL_CLEARED:
                    return 0L;

                // undo last stroke
                case SKN_STROKE_UNDO:
                    return 0L;

                default:
                    break;
            }
            return (MRESULT)TRUE ;

        case WM_COMMAND:
            switch (LOUSHORT(mp1))
            {
                ...
                // the user pressed the clear button
                case ID_CLEAR:
                    WinSendMsg(WinWindowFromID(hwnd, ID_SKETCH), SKM_DELETE_ALL_STROKES, 0L, 0L);
                    break;
                // the user pressed the undo button
                case ID_UNDO:
                    WinSendMsg(WinWindowFromID(hwnd, ID_SKETCH), SKM_UNDO_LAST_STROKE, 0L, 0L);
                    break;
                ...
            }
            break;
        ...
    }
}

```

Figure 5. Window procedure that controls aspects of the sketch control



```
USHORT SaveSketchToFile(HWND hwndDlg, CHAR *RecId)
{
    PSKETCHSTROKEDEF pStrokeData = NULL;
    ...

    // Get the total count of strokes in the sketch control.
    ulStrokeCount =
    (ULONG) WinSendMsg(WinWindowFromID( hwndDlg, ID_SKETCH), SKM_QUERY_CTL_STROKE_COUNT,
                      NULL, NULL);
    // if there is nothing in the sketch control erase the file
    if (ulStrokeCount < 1) {
        unlink(filename);
        return TRUE;
    }
    // Next, Open file to save Application Record, if first time - create it.
    rc = DosOpen(filename, &hf, &ulAction, 0, FILE_NORMAL, FILE_OPEN | FILE_CREATE, (ULONG)OPEN_ACCESS_READWRITE |
                OPEN_SHARE_DENYREADWRITE | OPEN_FLAGS_FAIL_ON_ERROR,
                (PEAOP2)NULL);
    ...
    ulSignatureSize = ulStrokeCount * sizeof(ULONG);
    // Malloc enough to save the lengths of each stroke.
    ulEachStrokeLength = (ULONG *) calloc(ulStrokeCount, sizeof(ULONG));
    // First stroke to the last stroke - Save each strokes' length in
    // the array which was malloced
    for (ulSaveIndex = 0; ulSaveIndex < ulStrokeCount; ulSaveIndex++) {
        ulEachStrokeLength[ulSaveIndex] =
            (ULONG) WinSendMsg(WinWindowFromID( hwndDlg, ID_SKETCH),
                              SKM_QUERY_STROKE_LENGTH,
                              // actual stroke number
                              MPFROMLONG(ulSaveIndex + 1),
                              NULL);
        ulSignatureSize += ulEachStrokeLength[ulSaveIndex];
    }
    // Write HEADER containing Stroke count and Sketch size.
    DosWrite(hf,
            &ulSignatureSize,
            sizeof(ULONG),
            &ulBytesWritten);

    DosWrite(hf,
            &ulStrokeCount,
            sizeof(ULONG),
            &ulBytesWritten);
    ...

    // Write ALL of the stroke's lengths to the file.
    DosWrite(hf, ulEachStrokeLength, sizeof(ULONG) * ulStrokeCount, &ulBytesWritten);
    ...
    // First stroke to the last stroke - Write each stroke to the file.
    for (ulSaveIndex = 0; ulSaveIndex < ulStrokeCount; ulSaveIndex++) {
        // Allocate a buffer (non-shared) based on that size.
        pStrokeData = malloc(ulEachStrokeLength[ulSaveIndex]);
    }
}
```

Figure 6. Code that saves sketch strokes to a file (continued on page 70)


```

...
// Get the Stroke information.
WinSendMessage( WinWindowFromID( hwndDlg, ID_SKETCH),
                SKM_QUERY_STROKE_DATA,
                MPFROMLONG(ulSaveIndex + 1),
                // actual stroke number
                MPFROMP(pStrokeData) );

// Write the Stroke to file.
DosWrite(hf,
         pStrokeData,
         ulEachStrokeLength[ulSaveIndex],
         &ulBytesWritten);

...
free(pStrokeData);
}
DosClose(hf);
free(ulEachStrokeLength);
return TRUE;
}

```

Figure 6. Code that saves sketch strokes to a file (continued from page 69)

```

USHORT ReadAndDisplaySketch(HWND hwndDlg, CHAR *RecId)
{
    PSKETCHSTROKEDEF pStrokeData = NULL;
    ...
    rc = DosOpen(filename,
                 &hf,
                 &ulAction,
                 0,
                 FILE_NORMAL,
                 FILE_OPEN, (ULONG)OPEN_ACCESS_READWRITE
                 | OPEN_SHARE_DENYREADWRITE | OPEN_FLAGS_FAIL_ON_ERROR,
                 (PEOP2)NULL)

    DosRead(hf, &ulSignatureSize, sizeof(ULONG), &ulBytesRead);
    DosRead(hf, &ulStrokeCount, sizeof(ULONG), &ulBytesRead);
    ...
    // Malloc enough to retrieve the lengths of each stroke.
    ulEachStrokeLength = (ULONG *)
    calloc(ulStrokeCount, sizeof(ULONG) );
    // Read All stroke lengths into array of lengths.
    DosRead(hf, (VOID *) ulEachStrokeLength,
            (sizeof(ULONG) * ulStrokeCount), &ulBytesRead);
    if ( ulBytesRead != (sizeof(ULONG) * ulStrokeCount) ) {
        DosBeep(500,200);
        DosClose(hf);
        free(ulEachStrokeLength);
    }
}

```

Figure 7. Code that reads stroke data from a file and redisplayes the strokes in a sketch control (continued on page 71)



```
        return(FALSE);
    }
    // First stroke to the last stroke - Read each stroke from the file
    for (ulStrokeIndex = 0; ulStrokeIndex < ulStrokeCount; ulStrokeIndex++) {
        // Allocate a buffer (non-shared) based on that size.
        pStrokeData = malloc(ulEachStrokeLength[ulStrokeIndex]);
        ...
        memset(pStrokeData, '\0', ulEachStrokeLength[ulStrokeIndex]);
        // Read the stroke from the file into the buffer.
        DosRead(hf,
                pStrokeData,
                ulEachStrokeLength[ulStrokeIndex],
                &ulBytesRead);

        // The following code is necessary for Stroke Data pointer fix ups after each read.
        pStrokeData->pStroke = (PSTROKEDATA) ( (PBYTE) pStrokeData + sizeof( SKETCHSTROKEDEF ));
        pStrokeData->pStroke->pXY = (PPOINTL)( (PBYTE) pStrokeData + sizeof( SKETCHSTROKEDEF ) +
                                                sizeof( STROKEDATA ));

        if (pStrokeData->pStroke->pAuxInfo) {
            pStrokeData->pStroke->pAuxInfo = (PAUXDATAINFO) ( (PBYTE) pStrokeData->pStroke->pXY
                + pStrokeData->pStroke->ulNumPoints * sizeof( POINTL ));
        }

        if (pStrokeData->pStroke->pAuxData) {
            pStrokeData->pStroke->pAuxData = (PAUXSTROKEDATA)( (PBYTE) pStrokeData->pStroke->pAuxInfo +
                sizeof( AUXDATAINFO )
                + sizeof( AUXDATADESC )
                * ( pStrokeData->pStroke->pAuxInfo->ulNumElements - 1 ) + pStrokeData->pStroke->ulNumPoints
                * pStrokeData->pStroke->pAuxInfo->ulAuxSize);
        }

        if ( ulBytesRead != ulEachStrokeLength[ulStrokeIndex]) {
            DosBeep(500,200);
            free(ulEachStrokeLength);
            free(pStrokeData);
            DosClose(hf);
            return(FALSE);
        }

        // Put the Stroke into the Sketch Control.
        rc = (BOOL)WinSendMsg( WinWindowFromID( hwndDlg, ID_SKETCH), SKM_ADD_STROKE,
            NULL,
            MPFROMP(pStrokeData) );

        ...
    }
    free(pStrokeData);
} // end for loop - completed reading all strokes

// Free the array of stroke lengths
free(ulEachStrokeLength);
DosClose(hf);
return TRUE;
}
```

Figure 7. Code that reads stroke data from a file and redisplay the strokes in a sketch control (continued from page 70)

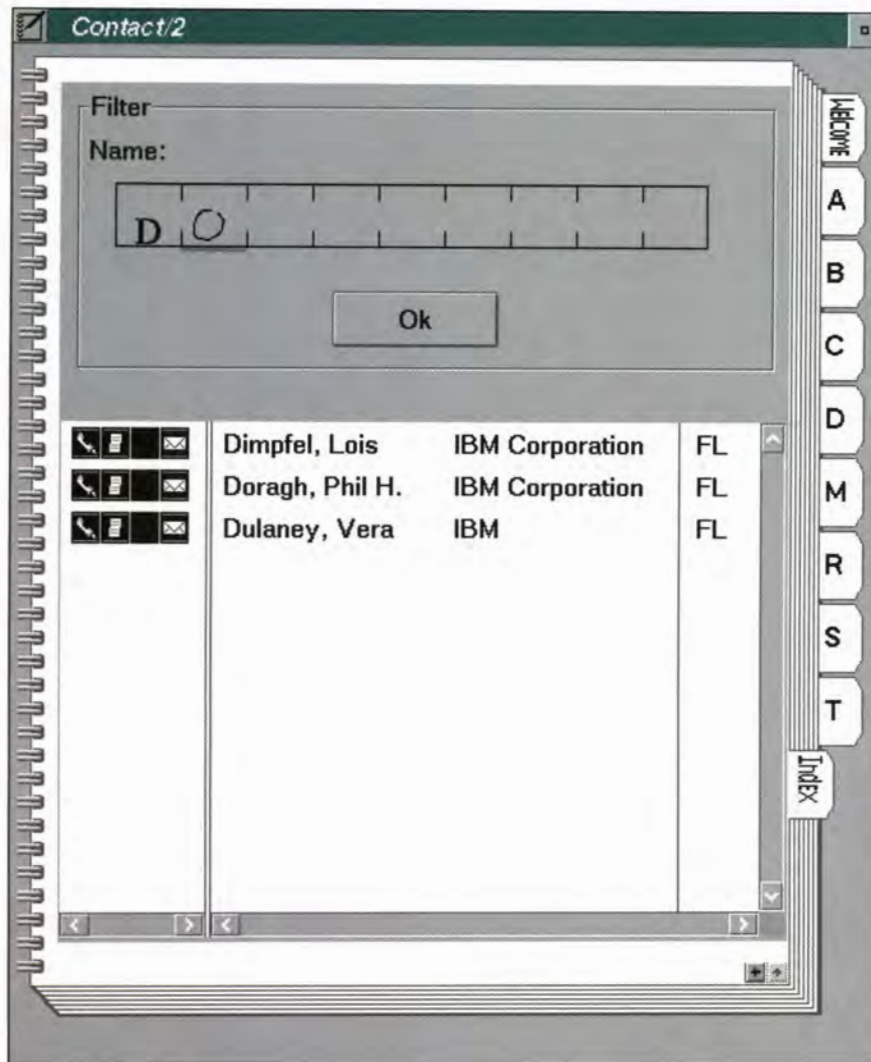


Figure 8. The index page

message along with the desired ink color. The code in Figure 5 shows how to use these messages.

THE INDEX PAGE

The **Index** page allows a user to see all of the entries in a book at a glance. Additional function has been added to this page by allowing a user to filter on all of the fields displayed for entries in the index view. The fields include first and last name, company, and state. Once a field is chosen, a combed handwriting control is displayed, allowing the user to hand write in the text. This is shown in Figure 8.

As each character of the text is recognized, the **Container** control

(list) filters out the entries that do not match the filter criteria. At any point, a user can tap on a name and the book will automatically turn to that entry's corresponding page. Also notice that the available types of communications for each entry in the book are indicated by the graphics symbol before each name. Figure 8 contains an example of the index page.

OTHER TIPS

Remember not to overdo handwriting entry recognition. Generally, you should support handwriting entry but not require it. Even though handwriting recognition is an exciting new technol-

ogy, it is still much simpler, quicker, and more reliable for a user to be able to tap with a pen rather than handwrite something. For example, it is better to create a list of selectable items than force the user to enter an item. Restrain your recognizable set of characters, if possible. For example, if your application is interested only in numbers, limit your recognizable character set to numbers. This will improve your recognition results. The capability to constrain the recognizable character set by allowing the programmer to specify various character sets or even individual characters is being considered for future releases of Pen for OS/2.

Future releases of Pen for OS/2 may also support handwriting recognition in single-line entry fields and multiline entry fields systemwide. In addition, ruled and unruled text entry may be available. If you know that your users will use the application with pen only (no keyboard), get rid of keyboard- and mouse-centric constructs such as double tapping, mouse button 2 scenarios, the concept of default buttons and highlights, pneumatics, and so on. Overall navigation should be achieved with mere taps of the pen. Another natural use of Pen Extensions is the enablement of allowing users to annotate on top of existing electronic documents.

Also, remember to replace key macros with gesture macros. Most of all, from the onset of your pen-centric application's design, remember that the user will now be using a pen. It is a new paradigm; users will want to interact differently with your application. It should feel more like pen and paper. As the user becomes more experienced with the pen environment, the application's constraints of the mouse and keyboard should become less

evident. A pen-centric application should be designed from the ground up for use with pen-based computers. The best way to achieve a pen and paper-like application is to design from the outside (look and feel) to the inside (code). The interface should require minimal computer knowledge on the part of the user.

Special thanks to Anthony E. Martinez for supplying the graphics work and ideas for the user interface representation. Also, I thank Janice Roberts for giving me the opportunity to create this application and her ideas for the index page.

Sarka J. Martinez, joined IBM in 1988 and is a senior associate programmer in the mobile applications software arm of the Mobile Pen and Speech (MVP) team in Boca Raton, Fla., which is part of the PSP division. Over the years, Martinez has worked on various advanced technology projects and is now involved in designing and implementing applications for IBM's PDA software solution. She received a B.S. in computer science at the State University of New York at Buffalo. Martinez has also received several patents for graphical user interface techniques.

REFERENCES

Code

Complete code blocks that are shown in this article are available on CompuServe's OS2DF2 Forum, OS/2 Developer section. Download file PENDEM.ZIP.

Products:

Pen for OS/2 Developer's Toolkit

Application Products Used:

CorelDraw 3.0

Microsoft PowerPoint 3.0

Adobe PhotoShop 2.5

IBM MultiMedia Extension

LOTUS cc:Mail

IBM Pen for OS/2 1.0

Publications:

OS/2 2.0 Technical Library Programming Guide Volume II (IBM Doc. S10G-6494).

OS/2 2.0 Technical Library Control Program Reference (IBM Doc. S10G-6263).

OS/2 2.0 Technical Library Presentation Manager Programming Reference II (IBM Doc. S10G-6265).

OS/2 2.0 Technical Library Presentation Manager Programming Reference III (IBM Doc. S10G-6272).

Burge, Thomas E. and Joseph Celi, Jr, *Advanced OS/2 Presentation Manager Programming*, New York, N.Y.: John Wiley & Sons Inc., 19XX. (ISBN 0-471-59198-X).

Pen for OS/2 User's Guide.

Pen for OS/2 Developer's Toolkit: Getting Started (71G2123).

Pen for OS/2 Programming Guide and Reference (71G2124).





Buyer's Guide

The staff of OS/2 Developer put together this special section to guide you through the various compilers and interpreters available for OS/2. The products described in this guide are based on surveys sent to vendors; they are provided as a service to you and are free to vendors. These listings do not represent an endorsement by OS/2 Developer. Although every effort has been made to ensure accuracy, we do not assume any responsibility for error or omission in this section. Please contact the companies directly for more information.

Compilers and Interpreters Buyer's Guide

ACUCOBOL INC.

Circle No. 101

Acucobol-85 2.1.1 for OS/2 2.0 and Acucobol-85 2.1.2 for OS/2 1.3 compilers offer machine-independent code that is portable, without recompiling, across different platforms, including AIX, AOS, MS-DOS, NetBIOS, OS/2, Ultrix, UNIX, XENIX, and VMS operating systems. The compilers include extensions for boxed and reverse video windows; pop-up windows; color; scrolling; machine-independent handling of file and directory names; file, record, and device locking; index file compression and encryption; system and terminal information; delete file; and reading of locked records. Acubol-85 also incorporates screen section support, DG-ICOBOL compatibility, a user-replaceable file system, and hot keys. Price: Please contact vendor.

Acucobol Inc., 7950 Silverton Ave., Ste. 201, San Diego, Calif. 92126, (800) 262-6585 or (619) 689-7220, fax (619) 689-7251.

ARITY CORP.

Circle No. 102

Arity/Prolog32 1.0 provides floating-point arithmetic, a menu-driven/windowed programming environment, a built-in editor, text string support, definite clause grammar support, and embedded interface to other programming languages. It provides support for concurrent Prolog execution, including a virtual database that supports two gigabytes of memory

and provides sequential, hashed, and b-tree indexing. It supports Microsoft Languages, Lattic C, and Assembly. Arity/Prolog32 contains a C compiler capable of handling C declarations and C expressions embedded with Prolog source code. The user can call C or Pascal functions from inside Prolog. The user can also call Prolog from C and Pascal and interface with dynamically linkable libraries. Price: \$2,450.00.

Arity Corp., Damonmill Sq., Concord, Mass. 01742, (800) 369-5622 or (508) 371-1243, fax (508) 371-1487.

BORLAND

INTERNATIONAL

Circle No. 103

Borland C++ 1.5 for OS/2 includes templates, exception handling, run-time type information, and ANSI string classes. It takes advantage of OS/2's multithreaded, multitasking environment and includes 32-bit optimizations and a Presentation Manager-hosted integrated device electronics. Other features include Borland BPMCC (custom controls) and Borland's ObjectBrowser for graphically displaying class hierarchy. Borland C++ for OS/2 also includes a free copy of Borland Turbo Assembler. Price: \$199.95; \$79.95, upgrade.

Borland International, 100 Borland Wy., Scott Valley, Calif. 95066-3249, (800) 331-0877 or (408) 431-1000, fax (408) 431-9119.

**CABOT SOFTWARE LTD.****Circle No. 104**

UCSD Pascal Compiler is integrated with Presentation Manager, allowing full Presentation Manager applications in Pascal. It is also integrated with the Power System development environment, which includes a set of programming tools and facilities: command menu with windowing and pop-up and pull-down menus; test and program editor; print utility to allow control of headers, pagination, format, spacing, and paper; library manager to add or remove generic code segments from code library; and file transfer. Applications generated by the Power System are stored in P-Code, a machine-independent programming standard that supports compact code files that can be smaller than the source text file. Price: \$299.00

Cabot Software Ltd. The Vicarage, Stoke View Rd., Bristol, U.K. +44-272-586644, fax +44-272-586680.

CALIFORNIA SOFTWARE PRODUCTS INC.**Circle No. 105**

BABY/4XX 1.5 enables the user to downsize AS/400 native code from a midrange system to the PC where it can be recompiled and used on a network or in multiuser or single-user mode. Running under OS/2, BABY/4XX allows the user to integrate AS/400 applications with DOS, Windows, and OS/2 software, such as word processors, spreadsheets, and databases. When used in conjunction with Novell NetWare, BABY/4XX provides client/server capability. Price: \$3,500.00.

BABY/AS 4.0 development systems permit the development and execution of IBM System/36 and AS/400 PRG II applications on IBM-compatible PCs. Running under OS/2, BABY/AS allows the user to integrate RGP II applications with DOS, Windows, and OS/2 software. BABY/AS can be used on a network or in multiuser or single-user mode. Price: \$3,500.00.

California Software Products Inc., 525 N. Cabrillo Park Dr., Santa Ana, Calif. 92701-5017, (800) 841-1532 or (714) 973-0440, fax (714) 558-9341.

CATSPAW INC.**Circle No. 106**

SPITBOL-386 1.28 is an implementation of the SNOBOL4 programming language. It features a vocabulary of string manipulation primitives for pattern matching and text analysis. Applications include conversion; reformatting; analysis of text; literature and music analysis; and program prototyping. Price: \$295.00.

Catspaw Inc., P.O. Box 1123, Salida, Colo. 81201, (719) 539-3884, fax (719) 539-4830.

COMPUTER ASSOCIATES INTERNATIONAL INC.**Circle No. 107**

CA-REALIZER 2.0A is a visual development tool that enables the development of cross-platform applications for individual, workgroup, and company-wide needs. This BASIC Construction Set is available on any system compatible with OS/2 2.1 or higher as well as Windows 3.1. CA-REALIZER helps the user create spreadsheets, charts, text editors, animation, graphics tablets, and forms from tools. Price: \$99.00.

Computer Associates International Inc., One Computer Associates Plaza, Islandia, N.Y. 11788-7000, (800) 225-5224 or (516) 342-5224, fax (516) 342-5734.

DATA ACCESS CORP.**Circle No. 108**

DataFlex 3.05 for OS/2 2.1 is an application development environment that includes a relational DBMS and an object-oriented, procedural fourth-generation language. This 32-bit character-cell tool allows users to share files with DataFlex for DOS or Windows users and interface with the DataFlex server. Price: \$795.00, single-user development; \$995.00+, multiuser.

Data Access Corp., 14000 SW 119th Ave., Miami, Fla. 33186, (800) 451-3539 or (305) 238-0012, fax (305) 238-0017.

DIGITALK**Circle No. 109**

PARTS Workbench 2.0 for OS/2 is a client/server integration tool that enables application development through assembly and reuse of application components. The Workbench comes with over 65 prebuilt



components, both visual and nonvisual. The user can write other components in Smalltalk/V, C, COBOL, or other languages. Price: \$1,995.00.

Smalltalk/V 2.0 for OS/2 is a 32-bit, object-oriented development environment for OS/2. It contains a set of tools for developing graphical, portable, CUA-compliant applications. Smalltalk/V includes a reusable class library, a push-button debugger, browsers, inspectors, and an incremental compiler. Price: \$995.00.

Digitalk, 5 Hutton Centre Dr., Santa Ana, Calif. 92707, (800) 922-8255 or (714) 513-3000, fax (714) 513-3100.

EDV—

KOHLNBACH

Circle No. 110

KIM 2.0 is a database developing system with a BASIC-like programming language and a dBase-compatible database engine. A free license runtime kit is available. KIM enables users to build OS/2 Presentation Manager programs. Price: \$150.00.

EDV Buero Kohlenbach Software-Entwicklung, 8502 Wintersdorf, Blumenstr. 58e, Germany - 90547 Stein, +49-911-688-9040-1, fax +49 9183-242.

GLANCE AG

Circle No. 111

Modula-2 to C Translator type M2CC 4.5 is a compiler system that translates Modula-2 applications into C. It enables the user to transfer Modula-2 programs from a VAX/VMS system to a DEC/Alpha machine, for example. The M2CC/C Modula-2 to C Translator also supports other systems with a C development environment, such as DOS, OS/2, and UNIX systems. The translator is also suitable for new and cross development of Modula-2 applications. Price: \$2,250.00; \$6,700.00 for workstations.

Glance AG, Gewerbestrasse 4, CH-8162 Steinmaur, Switzerland, +41 (1) 853-39-49, fax +41 (1) 853-08-09.

GPF SYSTEMS INC.

Circle No. 112

GpfRexx provides WYSISYG, visual, OS/2 Presentation Manager pro-

gramming with REXX. It allows the user to point and click to create CUA'91 applications. The Gpf Interface Builder handles the logic for managing events, windows, controls and their data, and for displaying help. Other features include multimedia, drag/drop, multitasking, SQL (DB2/2), and communications (APPC, CPI-C, EHLLAPI). No royalties. Price: Please contact vendor.

Professional Developer's Toolkit provides support for OS/2 2.x, OS/2 1.3.x, and Windows 3.0 and 3.1. It allows the user to point and click to create CUA'91-compliant GUIs. It generates C or C++ source, help source, and ancillary files. GpfTools enables manipulation and automatic documentation of Gpf designs. Price: \$1,440.00.

Gpf Systems Inc., 30 Falls Rd., P.O. Box 414, Moodus, Conn. 06469, (800) 831-0017 or (203) 873-3300, fax (203) 831-3302.

HOCKWARE INC.

Circle No. 113

VisPro/REXX 2.0 uses OS/2, Workplace Shell, and the REXX language as a visual programming environment. It provides the user with a development environment for creating, testing, debugging, modifying, and distributing OS/2 2.x GUI programs. VisPro/REXX is royalty-free. Price: \$99.00 to \$299.00+.

Hockware Inc., 315 N. Academy St., Ste. 100, Cary, N.C. 27515, (919) 380-0616, fax (919) 380-0757.

IBM

Circle No. 114

APL2 for OS/2 Entry Edition 1.0 is a programming language for application program developers and interactive users. It can be used to develop both commercial and scientific applications, "what-if" modeling, exploratory programming, interactive computing, decision support, and data analysis. It includes a Presentation Manager interface, graphics facilities, calls to other languages, and large work spaces. APL2 for OS/2 is compatible with the APL2 family of products, which run on IBM System/370 and System/390 with

CMS or TSO, the IBM RISC System/6000 with AIX/6000, the Sun SPARCstation with Solaris, and all personal computers with DOS. Part #89G1556. Price: \$185.00.

APL2 for OS/2 Advanced Edition is a programming language for application program developers and interactive users. It can be used to develop both commercial and scientific applications, "what-if" modeling, exploratory programming, interactive computing, decision support, and data analysis. It includes all of the features of the Entry Edition plus relational database access, tools for writing auxiliary processors, a performance monitor, and the ability to distribute data and processing among multiple APL2 systems connected through TCP/IP. APL2 for OS/2 is compatible with the APL2 family of products, which run on IBM System/370 and System/390 with CMS or TSO, the IBM RISC System/6000 with AIX/6000, the Sun SPARCstation with Solaris, and all personal computers with DOS. Part #89G1697. Price: \$650.00.

PL/I for OS/2 1.1 Professional Edition is an Operating System/2* (OS/2*) Version 2 based PL/I application development environment. It is designed for the experienced PL/I application developer who wants to have the power of the host language combined with the productivity of the PC tools. This compiler gives the PL/I application programmer the capability to write "client/server" applications in PL/I that access data on MVS, VM, AIX/6000, and OS/2. It is integrated with the DB2 family of products, with IMS Client Server/2, and with IBM's CICS family of products. The Toolkit—a separately priced feature—provides a set of valuable tools for visual building of screens for Presentation Manager. Part #10H7848. Price: \$750 (must order on or before Sept. 30, 1994, to receive this price).

PL/I for OS/2 1.1 Personal Edition is a full function Operating System/2* (OS/2*) Version 2 based PL/I application development environment. It

is designed for the PL/I application developer who wants to have the power of PL/I language available on a standalone PC or a small LAN.

IBM PL/I for OS/2 Personal Edition is ideal for programmers writing new PL/I applications that do not require access to mainframe data. Its applications can call Presentation Manager APIs, C-based applications or tools, and applications or tools written in REXX. The Toolkit—a separately priced feature—provides tools for visual building of screens for Presentation Manager. Part #10H7819. Price: \$299.00, Toolkit (part #1322966) is \$199.00.

IBM, 555 Bailey Ave., San Jose, Calif. 95141, (800) 426-2255, sales (408) 463-4871, technical (408) 463-4363, fax (408) 463-4820.

LESTEC PTY. LTD.

Circle No. 115 Modular And Integrated Design (MAID)

1.1 is an interface designer that helps the user create a repository of integrated working objects containing REXX event scripts. Features include multithreaded support; interdialog communication; a suite of subcommands and functions; and an external REXX and C interface. MAID provides support for database and host communications. Price: \$199.00.

Lestec Pty. Ltd., P.O. Box 1394, Dee Why, NSW 2099, Australia, +61 (2) 981-5925, fax +61 (2) 981-5925.

METAWARE

Circle No. 116
High C/C++ 3.2 for OS/2 is a "DirectToSOM" compiler. It enables the user to compile large programs and generate code. It is ANSI-com-

patible and provides support for C++ templates and exceptions. High C/C++ includes IBM's OS/2 Developer's Toolkit and DLL's to link into the WorkFrame/2 integrated environment. MetaWare offers free technical support and a money-back guarantee. Price: \$595.00; \$495.00, if user of another MetaWare compiler.

MetaWare, 2161 Delaware Ave., Santa Cruz, Calif. 95060, (408) 429-6382, fax (408) 429-9273.

MICRO DATA BASE SYSTEMS INC.

Circle No. 117
Object/1 3.0 is an object-oriented graphical development environment. Features include windows painter, source code management, and debugging tools. Price: \$4,000.00.



**Read and Write dBASE files
from C, C++, and now REXX!**

dbfLIB

Programmer's Library



New! dbfREXX

dbfREXX allows reading and writing dBASE files from your OS/2 REXX programs!

dbfLIB includes support for DOS, Windows, OS/2 and Windows NT all in one package!

**dbfREXX
FREE
with purchase of
dbfLIB!
Limited time**

**Sales (405) 360-3045
Tech Support (713) 537-0318
Visa and MasterCard accepted.**



dSOFT Development Inc.
4710 Innsbruck Drive
Houston, TX 77066

ASIAN OS/2 DEVELOPER SUPPORT



Get help porting OS/2 applications for Japanese, Chinese, or Korean.

Experienced guidance from the leading double byte character set OS/2 developer. Training or consulting on either an hourly or long term basis.



MicroBurst, Inc.

9035 Shady Grove Court
Gaithersburg, MD 20877

(301) 330-2995
Fax: (301) 330-8609
CompuServe: 70334.3616
Internet: 70334.3616 @
COMPUSERVE.COM.

Circle Reader Service Number 35

Circle Reader Service Number 36



GURU 3.1 is a development environment and relational database management system that offers information-processing tools. Within GURU, the user can access features such as a spreadsheet, business graphics, text processing, report generation, communications, and a language interface. GURU is compatible with SQL. It supports forward chaining, backward chaining, mixed chaining, multivalued variables, and fuzzy reasoning. Price: \$7,000.00.

Knowledge Man 3.1 combines a relational database management system and a fourth-generation language with a set of decision support tools. It includes KGL, an object-based, fourth-generation language and a KGL compiler. Binary large objects enable the user to store different types of information, including sounds and images, in variable-length records that the user can access with SQL queries. Price: \$1,500.00.

Micro Data Base Systems Inc., 1305 Cumberland Ave., P.O. Box 2438, West Lafayette, Ind. 47906, (800) 445-6327 or (317) 463-7200, fax (317) 463-1234.

MICRO FOCUS **Circle No. 118**
Micro Focus COBOL 3.2 contains a PC COBOL compiler and an interactive code debugger (Animator). Micro Focus has added 75 enhancements to this updated version, ranging from new syntax and increased performance to improved run-time management and usability enhancements. Price: \$750.00.

Micro Focus COBOL Workbench 3.2 is a suite of programmer productivity tools that allows the user to develop and maintain applications. Micro Focus COBOL Workbench includes Micro Focus COBOL Compiler, Micro Focus Toolset, and Micro Focus Operating System Extensions (OSX). Price: \$2,500.00.

Micro Focus, 2465 E. Bay Shore Rd., Palo Alto, Calif. 94303, (800) 872-6265 or (415) 856-4161, fax (415) 856-6134.

MICROWAY INC. **Circle No. 119**
NDP Pascal 4.4 for OS/2 implements the ANSI/IEEE standard 770X3.97-1983, a superset of Niklaus Wirth's Pascal. It includes several extensions from Berkeley 4.2 BSD Pascal and the British Standards Institute Level 0, a preprocessor, separate compilation of modules, and interfaces to the Microway C library. Price: \$595.00.

NDP C/C++ 4.4 for OS/2 is an AT&T 2.1-compatible compiler that includes NDP C, a dual-dialect compiler that supports two UNIX dialects (BSD 4.2 and AT&T 2.1), and the new ANSI C specifications. It features a function inliner, classes, single and multiple inheritance, constructors and destructors, and functions and operator overloading. Price: \$595.

NDP Fortran 4.4 for OS/2 offers standard global optimizations performed by a number of C compilers. It provides coprocessor support for the Weitek 3167/4167 and the Intel i860 and optimizations for the Pentium and 387/487 relating to storage allocation and inline execution of intrinsic functions. NDP Fortran is an F77 with VMS, VS, and MS extensions. It is compatible with the IBM C Set compiler, Toolkit, and WorkFrame. Price: \$595.00.

Microway Inc., Research Park, Box 79, Kingston, Mass. 02364, (508) 746-7341, fax (508) 746-4678.

PARCPLACE SYSTEMS INC. **Circle No. 120**
ObjectWorks Smalltalk is an object-oriented development environment for the development and delivery of color graphic applications in a heterogeneous computing environment. ObjectWorks includes Smalltalk lan-

guage, a set of integrated designed-for-objects development tools and a mature class library of over 400 types of portable objects. Price: \$2,495.00.

ParcPlace Systems, Inc. 99 E. Argues Ave., Sunnyvale, Calif., 94086, (408) 481-9090, fax (408) 481-9095.

PROMULA DEVELOPMENT CORP. **Circle No. 121**
Promula Fortran to Fortran Translator 5.02 transforms unstructured spaghetti Fortran code into well-structured Fortran code. It aids in cleaning up and upgrading old Fortran code. The translator supports most known Fortran dialects—standard or extended—including most Fortran 90 features. Price: \$1,500.

Promula Fortran to C Translator 5.02 is a portation tool that converts Fortran source code to C source code. It supports the Fortran 66 and Fortran 77 standards and many extended dialects (VAX, IBM, PRIME, SUN FORTRAN, and so on), including most Fortran 90 features. The C code it produces is portable and readable (maintainable). Price: \$1,500.00.

Promula Fortran to C++ Translator 5.02 converts FORTRAN code to C++ code. It supports standard Fortran 66, Fortran 77 and many extended dialects (VAX, SUN, PRIME, IBM VS), and most Fortran features. It produces C++ code that is readable and integrates with contemporary C++ OS/2 platforms—IBM C Set, Borland, Watcom, MetaWare, and Zortech. Price: \$1,500.00.

Promula Fortran Compiler 5.02 supports the Fortran 66 and Fortran 77 standards and many extended dialects (VAX, IBM, PRIME, SUN FORTRAN, and so on), including most Fortran 90 features. It compiles Fortran to C and is integrated with several OS/2 C/C++ compilers—

Watcom, Borland, Zortech, MetaWare, and IBM C Set. Price: \$1,000.00.

Promula Development Corp., 3620 N. High St., Ste. 301, Columbus, Ohio 43214, (614) 263-5454, fax (614) 263-5573.

QUERCUS SYSTEMS

Circle No. 122

Personal REXX 3.0 for OS/2 augments OS/2 REXX by including Quercus Systems' REXXLIB extensions for array handling, mathematical functions, interprocess communication, and general system services. In addition, it provides global variables; utilities for VM/CMS and VMS/TSO compatibility; and external data queue support. A variety of performance enhancements improve Personal REXX's speed. Price: \$175.00.

Quercus Systems, P.O. Box 2157, Saratoga, Calif. 95070, (408) 867-7399, fax (408) 867-7489.

REUSABLE SOLUTIONS INC.

Circle No. 123

Facets/4GL 3.0 is an object-oriented, fourth-generation language. This library of classes adds a collection of design tools to the Objectworks/Smalltalk development system. Through a series of on-screen forms, FACETS allows the user to generate screens, reports, menus, and other interface components. FACETS/4GL consists of five design tools: a schema designer, a forms designer, a report writer, a menu builder, and a help author. Price: \$2,000.00.

Reusable Solutions Inc., 1130 SW Morrison, #318, Portland, Ore. 97205, (503) 223-6892, fax (503) 223-4498.

SCIENCIA LTD.

Circle No. 124

Procyon Common Lisp 3.3.4 is an object-oriented development environment based upon Common Lisp and CLOS. Users can port applications to OS/2, Windows, and Macintosh

without code changes. Procyon is royalty-free. Price: \$1,800.00.

Scientia Ltd., St. John's Innovation Centre, Cowley Rd., Cambridge U.K., +44-223-42128, fax +44-223-42128.

TOWER TECHNOLOGY CORP.

Circle No. 125

TowerEiffel 1.3 is an object-oriented programming system that includes an Eiffel 3 compiler, a development environment, programming tools, and a set of reusable software components. The TowerEiffel compiler provides support for multilanguage development and interoperability between Eiffel, C, and C++. Price: \$1,295.00 for commercial license; \$295.00 for students.

Tower Technology Corp., 1501 Koenig Ln., Austin, Tex. 78756, (800) 285-5124 or (512) 452-9455, fax (512) 452-1721.

2500AD

SOFTWARE INC.

Circle No. 126

2500AD C Compiler has floating-point math, inline assembly language, ROM-able code, and memory bank-switching. The C Compiler includes an assembler, linker, librarian, macro preprocessor, simulator/debugger (if available), and object libraries. Full technical support and a 90-day money-back guarantee are included in the purchase price. Price: \$650.00, most versions.

2500AD Software Inc., 109 Brookdale Ave., Buena Vista, Colo. 81211, (800) 843-8144 or (719) 395-8683, fax (719) 395-8206.

VZ CORP.

Circle No. 127

VZ Programmer 2.3 is a programming environment that extends the function of the application by adding C++ code to any object in an object-oriented environment. This is a non-limiting programming environment that allows the programming of anything that can normally be pro-

grammed using C++. Price: \$2,000.00.

VZ Corp., 175 S. Main St., Ste. 700, Salt Lake City, Utah 84111, (800) 377-1090, (801) 595-1352, fax (801) 328-4404.

WATCOM INTERNATIONAL CORP.

Circle No. 128

Watcom C/C++ 10.0 is a C and C++ development system for DOS, Windows NT, Win32, OS/2 2.x, Novell NLS, and 16-bit DOS and Windows 3.x. It combines a compiler technology with a development environment and set of tools. Watcom C/C++ includes a GUI debugger, a C++ class browser, and a profiler. It provides support for C++ templates, exception handling, and Microsoft Foundation Class libraries. Price: Please contact vendor.

Watcom International Corp., 415 Phillip St., Waterloo, Ont. N2L 3X2, Canada, (800) 265-4555 or (519) 747-4971, fax (519) 747-4971.

XVT

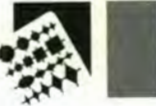
SOFTWARE INC.

Circle No. 129

XVT Development Solution for C (DSC) 4.0 pairs XVT-Design, a visual GUI layout, prototyping and code-generation tool, with the XVT Portability Toolkit, a thin-layered API that provides native look-and-feel and single-source portability to seven different GUIs and over 30 hardware platforms. Price: \$1,950.00-\$6,300.00.

XVT Development Solution for C++ (DSC++) 2.0 features XVT-Power++, an applications framework with features such as field validation, drag-and-drop, environment inheritance, nested views, geometry management, imaging, and the Rogue Wave data structures. Combined with the XVT Portability Toolkit, XVT-Power++ offers portability to many GUIs. Price: \$1,950.00-\$6,300.00.

XVT Software Inc., 4900 Pearl E. Cr., Boulder, Colo. 80301, (800) 678-7988 or (303) 443-4223, fax (303) 443-0969.



WE COVER OS/2

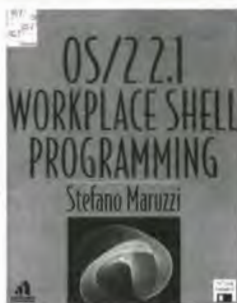
for Developers...

OS/2 2.1 WORKPLACE SHELL PROGRAMMING

Steve Maruzzi

"If C++ and Work-place Shell are in your career path, this book/software package should be in your hands."—*OS/2 Professional*

Packed with high-end tips and techniques, *OS/2 2.1 Workplace Shell Programming* offers a programmer's insight into OS/2 application development and shows C programmers how to implement a full OS/2 interface. A leading authority on OS/2, Maruzzi guides you through the process of building outstanding applications and interfaces with examples throughout the text and source code supplied on the accompanying disk. You'll get the low down on • the OS/2 system architecture • memory management • the graphics display interface • and much more. \$44, 744 pages, 0-679-79162-0



for Novices...

VAN WOLVERTON'S GUIDE TO OS/2, VERSION 2.1

Van Wolverton and Jim Meade

A critically acclaimed handbook and indispensable guide to OS/2 for beginners as well as for intermediate users who want to explore its latest features. Perfect for anyone who needs • straightforward, nontechnical explanations • a concise guide to OS/2 basics • and quick steps for running multiple applications. \$16, 144 pages, 0-679-74877-6



and Everyone in Between.

DVORAK'S GUIDE TO OS/2, VERSION 2.1

John C. Dvorak, David B. Whittle, and Martin McElroy

"A serious, detailed, dedicated study of OS/2 by authors who know the deepest secrets of a massive and powerful operating system."—*PC Magazine*

Dvorak's Guide to OS/2, Version 2.1 is a complete command reference that thoroughly explains all OS/2 commands. An extensive tips and tricks section will turn you into a power user. The authors provide important resources to help you solve any problem, plus a complete listing of worldwide OS/2 computerized bulletin boards. \$45, 816 pages, 0-679-74648-X



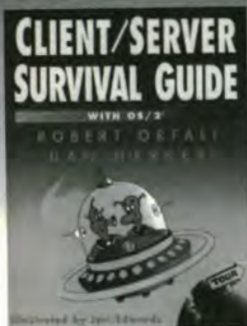
ADVERTISER INDEX

Reader Service	Company Name	Page
44	Borland International.	COV4
33	Creative Systems Programming	61
12	ColoradOS2	39
17	Development Technologies, Inc.	49
35	dSoft Development	77
25	Essex Systems, Inc.	51
14	FTG Data Systems.	45
7	Gpf Systems Inc.	15
5	Great Lakes Software.	9
	IBM Software Solutions.	11
16	Impact Software	49
20	Indelible Blue, Inc..	50
23	IRMware Services	51
2	Intersolv	1
22	Keller Group	51
42	Levine Computer Consultants	94
10	Metaware.	23
26	MHR Software & Consulting.	47
36	Micro Burst, Inc.	77
3	Micro Focus	2
32	Mitnor Software	59
13	On-Line Data.	41
11	One Up Corp.	27
21	Op-Tech Data Processing.	51
19	Perez Computing Service.	50
9	Poet Software	19
4	Programmer's Paradise.	4
40	Quadron	86
37	Random House	80
	Rexx Report	18
31	Rhetorex Inc.	58
43	Rimstar Technology	COV3
30	SE International	55
39	Softbridge, Inc.	86
29	SofTouch Systems	54
	Software Development 94 East.	33
41	The Object People	87
8	Token Technology.	17
24	Token Technology.	51
18	Team Development Corporation	50
38	Van Nostrand Reinhold.	81
34	Warp Speed	62
1	Watcom C for IBM PC's	COV2

LOOK FOR THEM IN THE COMPUTER
SECTION OF YOUR LOCAL BOOKSTORE.



VNR KNOWS OS/2®



CLIENT/SERVER SURVIVAL GUIDE WITH OS/2 2.1 by Robert Orfali and Dan Harkey. A sweeping tour of client/server systems and software. \$39.95 0-442-01798-7 IBM# SR28-5494

CLIENT/SERVER PROGRAMMING WITH OS/2 2.1, 3rd Ed., by Robert Orfali and Daniel Harkey. "Absolutely one of the most impressive, useful, best-written computer books I've ever seen."—Will Zachman. \$39.95 0-442-01833-9 IBM# G325-0650-02

THE GUI-OOUI WAR: Windows vs. OS/2 by Theo S. Mandel. "A book that should be forced on every Windows and OS/2 developer working today. An exceptionally clear, readable guide. If you want to make a good program great, there's no better starting place."—OS/2 Professional \$29.95 0-442-01750-2 IBM# SR28-5251

OS/2 FUNCTIONS QUICK REFERENCE LIBRARY 1: Win Functions by Nora Scholin. \$19.95 0-442-01897-5

C AND C++ PROGRAMMING IN THE OS/2 ENVIRONMENT by Mitra Gopaul. \$39.95 0-442-01240-3 IBM# SR28-4404

OS/2 2.1 APPLICATION PROGRAMMER'S GUIDE by Jody Kelly, Craig Swearingen, Dawn Bezviner and Theodore Shrader. \$34.95 0-442-01736-7 IBM# SR28-5449

OS/2 2.1 REXX HANDBOOK: Basics, Applications, and Tips by Hallet German. A great reference book that contains info not found in any other REXX books. \$29.95 0-442-01734-0 IBM# SR-5250



THE OS/2 2.1 CORPORATE PROGRAMMER'S HANDBOOK by Nora Scholin, Mark Sullivan, and Robin Scragg. "A solid job. Introduces readers to programming terms and concepts. Explanations are lucid and reading relevant sections provides you with the basics you need."—OS/2 Professional. Includes disk. \$39.95 0-442-01598-4 IBM# SR28-4390

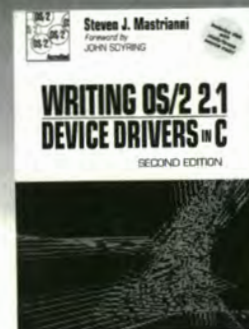
WRITING OS/2 2.1 DEVICE DRIVERS IN C, 2nd Ed., by Steven Mastrrianni. "Should give any C programmer with a vague knowledge of OS/2 and PC hardware a headstart in writing OS/2 device drivers in C."—International OS/2 User Group Magazine. Includes disk. \$39.95 0-442-01729-4 IBM# SR28-4392

THE SHELL COLLECTION: OS/2 2.X Utilities by Steven Levenson. A great source of shortcuts, tips and productivity boosters for Windows and DOS. Includes disk. \$29.95 0-442-01585-2 IBM# G362-0049

USING WORKPLACE OS/2: Power User's Guide to IBM OS/2 Version 2.1 by Lori Brown and Jeff Howard. The authors are the principal designers of the Workplace Shell! Includes disk. \$24.95 0-442-01590-9 IBM# SR28-4391

THE OS/2 2.0 HANDBOOK: Applications, Integration, and Optimization by William H. Zack. "Should be within fingertip reach of anyone working with OS/2. IBM would do well to include a copy in every package of OS/2."—OS/2 Professional \$34.95 0-442-01234-9 IBM# G362-0009

OS/2® is a registered trademark of the IBM Corporation.



OS/2 2.X NOTEBOOK: The Best of IBM OS/2 Developer, edited by Dick Conklin. \$36.95 0-442-01522-4 IBM# G362-0015

NOW THAT I HAVE OS/2 2.1 ON MY COMPUTER, WHAT DO I DO NEXT?, 2/E. by Steven Levenson. "Easy to read, loaded with examples, a friendly reference source."—Journal of The Chicago Computing Society. \$22.95 0-442-01832-0 IBM# G362-0008-01

COMPREHENSIVE DATABASE PERFORMANCE FOR OS/2 2.0's Extended Services by Bruce Tate, Tim Malkemus and Terry Gray. \$41.95 0-442-01325-6 IBM# G362-0019

LEARNING TO PROGRAM OS/2 2.0 PRESENTATION MANAGER BY EXAMPLE: Putting the Pieces Together by Stephen Knight. Includes disk. \$41.95 0-442-01292-6 IBM# G362-0011

THE COBOL PRESENTATION MANAGER PROGRAMMING GUIDE by David M. Dill. \$41.95 0-442-01293-4 IBM# G362-0010

OS/2 PRESENTATION MANAGER GPI GRAPHICS by Graham C. E. Winn \$42.95 0-442-00739-6 IBM# G362-0005



Order directly from the publisher by calling 1-800-544-0550. To order from IBM call 1-800-568-2694



Van Nostrand Reinhold
115 Fifth Avenue, New York, New York 10003
1-800-544-0550 73373.64 @compuserve.com



New Tools for OS/2



TOOLS

Techbridge Builder. This GUI, client/server/ object-oriented application development environment for the IBM OS/2 operating system allows anyone, from experienced users to professional developers, to create sophisticated workplace applications involving drag-and-drop operations and enterprise-wide database access in object-oriented manner. Developers using TechBridge Builder simply paint the majority of an application into existence with visual programming tools and code the rest in standard COBOL or SQL.

For the advanced developer, TechBridge Builder provides object-oriented extensions to the COBOL language support.

TechBridge Technology Corp.
Phone: (416) 222-8998, Fax (416) 222-0168

XDB-Server 4.0. XDB-Server 4.0 provides enhanced server functionality for distributed client/server environments through a rich set of development tools, including optimized query performance, server-to-server connectivity, and administration tools. XDB-Server 4.0 achieves faster response times through performance-tuning options, allowing the user to fine-tune the database server. Users also have the flexibility of remotely accessing data while not robbing the enterprise of consistent system administration. Database administrators and developers familiar with DB2 3.1 can develop and target applications on a client/server LAN using XDB-Server 4.0. Of particular note to OS/2 users, XDB-Server 4.0 has a 32-bit, multithreaded architecture.

XDB Systems Inc.
Phone: (301) 317-6800 ext. 146,
Fax: (301) 317-7701

dbfREXX 1.0. This DLL for OS/2 2.1 extends the built-in REXX language with commands for manipulating dBase files, including DBF, DBT, and NDX. According to the company, REXX users will be able to tap into simple database management without all the overhead of a full-blown client/server DBMS.

dSOFT Development Inc.
Phone: (405) 360-3045 or (713) 537-0318

Watcom SQL. Watcom SQL network servers for OS/2, which are available in single-user and multiuser versions, benefit from the advantages of the 32-bit OS/2 platform and exploit multitasking capabilities to run concurrently with other applications, eliminating the need for a dedicated database server. Watcom SQL uses the Open Database Connectivity standard for SQL database connectivity as its functional API, providing interoperability with a range of front-end tools. It also offers application programming features such as bidirectional, scrollable, updatable cursors; updatable multitable views; binary large objects; self-tuning, cost-based query optimization; database compression; and multinational character set support.

Watcom International
Phone: (519) 886-3700

UNIFACE. The company has announced the availability of UNIFACE tools that are optimized for IBM's DB2/2 and DB2/6000 rela-



tional database management systems. A new interface links the UNIFACE development system with DB2/2, DB2/6000, and host-based servers. The interface aims to provide transparent and simultaneous read/write access to DB2/6000 databases under AIX and DB2/2 databases under OS/2 and will support host-based DB2 security, error handling, data integrity and recoverability, and all DB2 data types. UNIFACE applications are designed to access many relational and non-relational data sources, including DB2/2, Rdb, INGRES, ORACLE, Sybase, and Microsoft SQL server.

Uniface Corp.
Phone: (510) 748-6025

ESL Workbench 4.0. This OS/2-based integrated development environment enables developers to rapidly create simultaneous multiple-language client/server applications. New CUA extensions have been added, including spin button, slider control, and drag and drop to enhance graphical user interface development. Integration with Intersolv's PVCS multideveloper support allows teams of developers to simultaneously work on an application by version control and creation of archive files. ESL Workbench 4.0 also features the ESL Wizard, a visual programming tool that provides reusable code libraries and dynamic code generation.

Easel Corp.
Phone: (617) 221-2100, Fax: (617) 221-3099

AlertVIEW. The company has announced AlertVIEW for Lotus Notes, which is installed on OS/2 Lotus Notes servers to report replication, server, security, statistics, mail, and communication events. This agent detects errors that are not reported in the standard error log and also reports errors from remote Notes servers connected using dial-up services.

Shany Inc.
Phone: (415) 694-7410, Fax: (415) 694-4728

AutoTester 2.0. Just released for the OS/2 Presentation Manager operating system, AutoTester 2.0 is a testing and verification tool for graphical user interface applications. The software provides word-processing type editing of tests, including familiar features such as cut/copy/paste, find/replace, and adjustable fonts. The primary focus of the AutoTester 2.0 release is the introduction of AutoCommand, which generates a documented automated test instead of recording only a series of keystrokes and mouse events.

AutoTester Inc.
Phone: (800) 328-1196 or (214) 368-1196

Subpanes/v. The company has added 20 new controls to this library of controls for Smalltalk/V and WindowBuilder Pro/V. The new controls include columnar list box, Table Pane, Bitmap Pane, Bitmap Button, Sliders, Gauges, 3-D Frames, HierarchicalListBox, DateEditor, TimeEditor, and NumberEditor.

Objectshare Systems Inc.
Phone: (408) 970-7280, Fax: (408) 970-7282

OS/2 CALENDAR

July 19-22	OS/2 World, Santa Clara, Calif. (415) 905-2354
Oct. 4-6	Software Development East '94, Washington, D.C. (415) 905-2414
Oct. 30-Nov. 4	ColoradOS/2, Colorado Springs, Colo. (719) 576-5003
April 1995	OS/2 Development '95 Copenhagen, Denmark (212) 626-2275



Pen Systems

Touch-screen technology has made it possible to deploy automation where it was previously thought impossible. Doing so has required some rethinking of the user interface. To enhance usability, speed and ease of use were given paramount consideration. But it all relies upon hardware to provide an intuitive pen or touch "feel." By the CUBS TEAM—ROLANDO ARQUIZA, JERRY BAYER, LARRY GAVIN, BRIAN NORQUIST, AND JIM ZEIDEL

OS/2 Survives the Pits!

They said it couldn't be done. "Impossible!" "Not in our trading pits." But they were wrong. For well over a year now, a number of floor brokers in Chicago have been using OS/2 with pen based technology, right in the heart of the Chicago Mercantile Exchange's (CME) future trading pits.

CHICAGO'S CUBS

The system the brokers are using is called CUBS, the CME Universal Broker Station. This system, in conjunction with its front-end processor, allows firms located around the world to route orders directly to brokers in the pit. The system also provides the broker with deck management, execution, and endorsement capabilities that instantly inform the firm whenever a trade has taken place. Compared to the current manual system, CUBS offers a number of advantages as it takes recent computing technology where it has never gone before—the pits.



(l-r) Brian Norquist, Rolando Arquiza, Larry Gavin, Jerry Bayer

LIFE IN THE PITS

It was easy, however, to see why the skeptics felt the way they did; the

Chicago-styled open outcry method of trading is very physically and mentally demanding. If you have not seen these trading pits on television or in the movies, it may be difficult to imagine what it's like.

Picture yourself in a jam-packed elevator, shoulder to shoulder, chest to back with people. Add the element of chaotic market swings, where fortunes can be made or lost in a matter of minutes. And remember that you are in direct competition with everyone around you in getting off your order. That means being seen and heard first. You have to stay alert and act fast—very fast.

This is the "major league" of trading. On some days, the Chicago Mercantile Exchange executes over two million contracts. That represents over two trillion dollars in underlying contract value. That's more than the entire U.S. government budget for a year changing hands in a single day! This is how worldwide businesses insure themselves against market risk. This is where the business world turns to secure some stability for decision making. And at this level, you can be sure CME impacts your life in untold ways, even if you have never heard of futures.



INTRODUCING OS/2 PEN TECHNOLOGY

It is in this environment that OS/2 with touch-screen technology has found a home. Why? Because this technology can stay alert and act very fast. This technology is the preemptive design of OS/2 coupled with its multitasking/multithreading capabilities. All of which is presented to the user via a natural-feeling, responsive touch screen. Obviously, the traditional keyboard and mouse cannot be used effectively in this environment. Without pen technology, this application would never have been conceived.

It's that high-pressured, time-critical, physically draining environment that demands an application that is intuitive to use—right down to the feel of the pen on the screen. Anything less would be technology hindering, not helping, the broker.

For example, a lesser operating system might pause momentarily while simultaneously:

- Receiving orders from as many as 80 firms
- Updating the window of open orders
- Displaying a continuous stream of market quotes
- Printing an endorsement ticket.

If a pause occurs as the broker tries to register a trade, the broker may try again, entering a second double-click. So when the system returns to process the user input queue, it might actually kick off two trades. This would cause confusion at the worst possible time—when the broker is most busy. At the very least, it might require the broker to wait, distracting the broker's attention. All of this is unacceptable: the broker's livelihood depends on paying complete attention to the trading pit, not on wrestling with some piece of equipment.

CUBS avoids these pitfalls. Using OS/2's preemptive operating system,

user input is given highest priority, returning the necessary visual feedback immediately, while the other multithreaded tasks are efficiently completed simultaneously.

IT'S PERFECT!

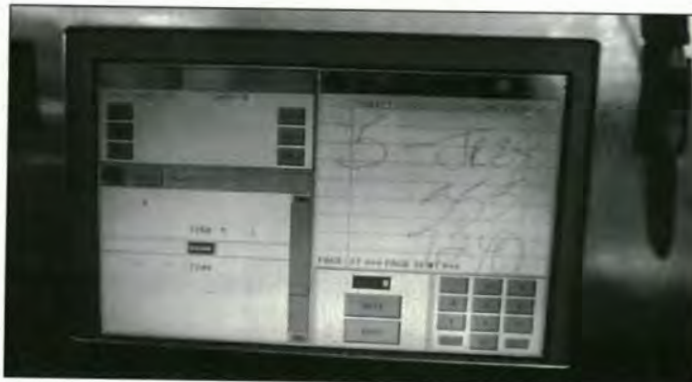
The results speak for themselves. According to Ed Zegar, vice president of GSA and one of CME's CUBS brokers, "It's perfect!" But to get that kind of response from our users required not only OS/2 and IBM's touch screens but also a close working relationship with users in order to fine-tune the application.

AN ENDORSEMENT FOR HANDWRITING

At first the system was designed so that push buttons were used for everything. But during mock trading sessions, we soon found that—surprise!—the traders were accustomed to using a pen as a pen, not just as a pointing device. They wanted to write out their endorsements, not use push buttons. An endorsement consists of:

- The actual execution price
- The opposite broker(s)
- The opposite broker's house
- Some other information.

However, a push-button approach does not duplicate the nuances of written endorsements. Every written endorsement is slightly different. Thus, if a broker



A broker's CUBS screen used in the pits



is interrupted in the middle of writing an endorsement—say, to execute a trade—the broker could look at the first stroke on the screen and recall the transaction along with related information.

Looking at nicely printed text generated from pushed buttons does not invoke the same level of recall. One transaction looks similar to another. This recall capability is important since it allows brokers to be interrupted many times and to queue endorsements in their heads while more immediate tasks are addressed. The ability to do this lets brokers act very fast when trading.

THE NATURAL WAY

Capturing handwriting is more like the way brokers have been doing their jobs for years. It works,

```
// BUTTON DOWN CODE
PFNWP      buttonProc = NULL;

/*****
 *
 * FUNCTION: RegisterButtonDwnClass
 * The function registers a window class that is a subclasses of the
 * WC_BUTTON window proc. It changes the buttons response to
 * WM_BUTTON1DOWN and WM_BUTTON1UP. It changes the button proc to
 * produce a WM_COMMAND on the location of WM_BUTTON1DOWN not on
 * WM_BUTTON1UP.
 *
 * *****/
BOOL WINAPI RegisterButtonDwnClass( HAB hab )
{
    CLASSINFO classinfo;
    if ( buttonProc == NULL )
    {
        if ( WinQueryClassInfo( hab, (PSZ) WC_BUTTON, &classinfo ) )
        {
            buttonProc = classinfo.pfnWindowProc;
            classinfo.flClassStyle &= CS_PUBLIC;
        }
    }
}
```

Figure 1. Button down code (continued on page 88)

We'd like to dispel a couple of myths about client/server testing.

Myth #1: Client/Server testing is easy.

New development tools make building client/server applications easier than ever. The problem is, they're harder to test, and you just can't stand up to them with manual testing methods. **You need ATF, the only testing product designed for client/server.**

Myth #2: Client/Server testing is impossible.

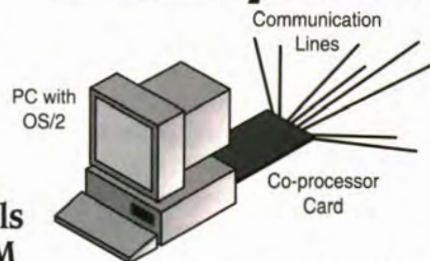
Testing client/server applications can be daunting. The problem is, your tests need to replicate real-world conditions, and you just can't do that with a stand-alone testing product. **You need ATF, the only testing product designed for client/server.**

The Softbridge Automated Test Facility's unique architecture makes it the only product for true client/server testing of OS/2 and Windows apps. To learn more about ATF, call 617-576-2257. (Or FAX 617-864-7747.)

If you're coming to *OS/2 World* in Santa Clara this July, stop by Booth 615 for an ATF demo. **There's only one way to test client/server applications, and that's with ATF.**

Softbridge, Inc. ▲ 125 CambridgeParkDrive ▲ Cambridge, MA 02140

Cut your development time



Quadron's OS/2 development tools team up with IBM

ARTIC co-processor cards to give you flexibility, communication power, and development speed.

Use Quadron software and ARTIC cards to gain extra ports and enhance host PC performance by moving protocol processing to the cards. Develop standard and custom communications for async, BISYNC, HDLC, X.25, LAP-B, or your own special protocol. And if you need more than one protocol, no problem—just run them at the same time on the same card.

Programmers feel right at home with our software. Just program in standard C, and use our messaging API to link OS/2 threads with

tasks on the card. Then check it out with our symbolic debugger and real-time analysis tools in OS/2 windows.

Our easy-to-use software serves applications as diverse as telephone switching, financial transactions, satellite communications, and process control. If any of this sounds like a match for your needs, call us today.

Quadron

209 East Victoria Street
Santa Barbara, CA 93101
Fax: 805-966-7630
805-966-6424



Trademarks are the property of their respective owners.

and they want to keep doing it that way. Technology should serve the user, not the other way around. So, we changed the application to capture their handwriting. The brokers love it.

IT'S GOTTA BE COLOR

Color also plays a large role. At first, only black and white screens were used. They worked, but as soon as the brokers saw the color units, they no longer wanted the black and white screens. Color gives important visual feedback. The user does not have to read words on the screen. Blue means a buy order, and red means a sell order. Brokers do not have to read the words to know what side of the market they are on. This saves time when they are entering and filling orders.

On the black and white units, this critical distinction is not as obvious. Color, therefore, lets brokers act quickly and avoid costly mistakes.

LET YOUR FINGERS DO THE TRADING

On some of the candidate touch screens, a unique and dedicated pen was required. Brokers could not use those systems if they lost their pens or if their pens stopped working for some reason. To guard against this, the units deployed are both pen- and finger-sensitive. If brokers lose their pens, they can use their fingers. (Sometimes in the pits, brokers have to be careful about losing fingers as well!)

MAKE THOSE BUTTONS BIG!

One important aspect of the user interface was the size of the but-

tons; they had to be large enough for a gorilla's finger! Although this eats up valuable screen geography, it has the advantage of providing a large landing area. Users appreciate this even when they are using a pen.

To make it even easier to use the touch screen—especially when brokers are being jostled around during fast markets—the buttons had to be subclassed to register a touchdown even though the broker's pen or finger slid off before it was picked up. This is not standard CUA protocol, but in this environment, it's what works. Figures 1 and 2 show how that subclassing was done.

BUTTONS FOR BARS

To further the idea of large, easy-to-hit touch areas, we used a row



**MAY THE
SOURCE
BE WITH
YOU**

Don't let the dark forces of ignorance defeat you. Tap into the free CONSUMER INFORMATION CATALOG -- the source for more than 200 free and low-cost government publications on cosmic topics such as health, travel, nutrition, careers and government benefits. Just send your name and address to:

**Consumer Information Center
Department Source
Pueblo, Colorado 81009**

Transitioning to Smalltalk technology?
Introducing Smalltalk to your organization?
Travel with the team that knows the way...

The Object People
"Your Smalltalk Experts"

SMALLTALK

Education & Training • Smalltalk/V (Windows, OS2, Mac) • VisualWorks • Smalltalk for Cobol Programmers • VisualAge • ENVY/Developer • Analysis & Design • Project Management • In-House & Open Courses	Project Related Services • Consulting & Mentoring • Rapid Prototyping • Custom Software Development - Legacy Systems - GUI's, Databases - Client-Server
--	---

The Object People Inc. 509-885 Meadowlands Dr., Ottawa, Ontario, K2C 3N2
Telephone: (613) 225-8812 FAX: (613) 225-5943

Smalltalk/V is a registered trademark of Digitalk, Inc.
VisualWorks is a trademark of ParcPlace Systems Inc.

VisualAge is a registered trademark of IBM.
ENVY is a registered trademark of OTI Inc.

of buttons painted across the top of the windows to support various activities instead of the standard action bar. Instead of the standard action pull down menu, a column of easy-to-hit buttons appears. Figure 2 shows how we did this.

OTHER THINGS LEARNED ALONG THE WAY

During the mock training sessions, we were surprised to learn that some people have a difficult time double-clicking with a pen. Others seem to do it naturally. So the system was designed to have large "buy" or "sell" buttons on it for those who would prefer the one-button, one-click approach.

HAVE IT YOUR WAY

We used command-line parameters for situations that do not change dynamically and user-definable buttons for those situations that do. For example, our keypad has user-definable buttons for quantities; touching a 1 and then a 0 button is not as fast as a single touch of a 10 button. Later, when brokers are trading 25 lots more often, we can reset that user-definable button to 25 on the fly.

In another example, our numeric keypad has a command-line parameter for left- or right-

```

        if ( !WinRegisterClass( hab,
                                WC_BUTTONDOWN,
                                (PFNWP) CubButtonDwnSubProc,
                                classinfo.flClassStyle,
                                classinfo.cbWindowData))
        {
            return FALSE;
        }
    } /* end if */
    else
        return FALSE;
    } /* end if (buttonProc == NULL) */

    return TRUE;
}

/*****
 *
 * FUNCTION: CubButtonDwnSubProc
 * Intercepts WM_MOUSEMOVE messages in order to keep the button state
 * highlighted after a button down even if the mouse pointer goes
 * outside of the buttons boundaries. This will cause a command message
 * to be generated always on a button up, even if it occurs outside the
 * window boundary.
 *
 *****/
MRESULT EXPENTRY CubButtonDwnSubProc( HWND hwnd, USHORT message,
                                       MPARAM mp1, MPARAM mp2 )
{
    switch ( message )
    {
        case WM_MOUSEMOVE:
            return (WinDefWindowProc( hwnd, message, mp1, mp2 ));

        default :
            return ((*buttonProc)( hwnd, message, mp1, mp2 ));
    }
}

```

Figure 1. Button down code (continued from page 86)

```

// BUTTON MENU code

#define FID_BUTTONMENU 0x8016 /* numerically greater than FID_CLIENT */
PFNWP buttonMenuFrameProc; /* original procedure from the subclass
 * procedure of the button menu */
INT buttonMenuHeight; /* used in CubFormatFrameSubProc */
PFNWP formatFrameProc; /* window procedure of standard frame
 * window as subclassed for the cub
 * button menu */
HWND hwndButtonMenu; /* Frame Button Menu Window handle */
SIZEL sizlButton; /* default button size */
SIZEL sizlButtonSpacing; /* space between buttons */

/*****

```

Figure 2. Button menu code (continued on page 89)


```

* FUNCTION: CubCreateButtonMenu
* This Function creates a button menu that returns all WM_COMMANDs
* to its owner, listing a menu as the source of the command.
*****/
HWND EXPENTRY CubCreateButtonMenu( HWND hwndParentFrame, HWND hwndOwner )
{
    COLOR        backgroundColor;
    FRAMECDATA   fcdata;
    FONTMETRICS  fm;
    HAB          hab;
    HPS          hps;
    SIZEL        sizlPS;

    hab = WinQueryAnchorBlock( hwndParentFrame );
    formatFrameProc = WinSubclassWindow( hwndParentFrame,
                                          (PFNWP)CubFormatFrameSubProc );

    fcdata.cb = sizeof( FRAMECDATA );
    fcdata.flCreateFlags = FCF_NOBYTEALIGN | FCF_BORDER;
    fcdata.hmodResources = 0;
    fcdata.idResources = FID_BUTTONMENU;
    hwndButtonMenu = WinCreateWindow( hwndParentFrame,
                                      WC_FRAME,
                                      NULL,
                                      WS_VISIBLE | WS_CLIPSIBLINGS,
                                      0, 0, 0, 0,
                                      hwndOwner,
                                      HWND_TOP,
                                      FID_BUTTONMENU,
                                      &fcdata,
                                      NULL );

    if ( hwndButtonMenu != (HWND)0L )
    {
        buttonMenuFrameProc = WinSubclassWindow( hwndButtonMenu,
                                                  (PFNWP)CubButtonMenuSubProc);

        backgroundColor = WinQuerySysColor( HWND_DESKTOP, SYSCLR_SHADOW, 0L );
        WinSetPresParam( hwndButtonMenu,
                        PP_BACKGROUND_COLOR,
                        (ULONG) sizeof( COLOR ),
                        &backgroundColor );

        hps = WinGetPS( hwndButtonMenu );
        GpiQueryFontMetrics( hps, sizeof( FONTMETRICS ), &fm );
        avgUpperCharWidth = (INT) fm.lEmInc;
        /* use default presentation space page size */
        sizlPS.cx = 0;
        sizlPS.cy = 0;

        /* use standard english units for sizing, but convert into pixels for
         * speed, in units of 0.001 inches */
        GpiSetPS( hps, &sizlPS, PU_HIENGLISH );

        /* sizlButton is considered to be the minimum size for a finger touch
         * on the screen. It is used to set the size of the button menu. */
        sizlButton.cx = 597;
        sizlButton.cy = 479;
        GpiConvert( hps,
                   CVTC_PAGE,
                   CVTC_DEVICE,
                   1L,
                   (PPOINTL) &sizlButton );
    }
}

```

Figure 2. Button menu code (continued on page 90)


```

        sizlButtonSpacing.cx = 33;
        sizlButtonSpacing.cy = 33;
        GpiConvert( hps,
                    CVTC_PAGE,
                    CVTC_DEVICE,
                    1L,
                    (PPOINTL) &sizlButtonSpacing );
        WinReleasePS( hps );

        buttonMenuHeight = (SHORT) (sizlButton.cy +
                                     (2 * sizlButtonSpacing.cy));
    }
    return hwndButtonMenu;
}

/*****
 * FUNCTION: CubButtonMenuSubProc
 * Although this is a subproc, it performs all the basic logic for
 * button menus. It lets the default frame proc handle the basic stuff.
 *****/
MRESULT EXPENTRY CubButtonMenuSubProc( HWND hwnd, USHORT message,
                                       MPARAM mp1, MPARAM mp2 )
{
    HWND      hwndOwner;
    HWND      hwndParent;
    switch ( message )
    {
        case WM_COMMAND:
            /* make sure the source is menu */
            hwndOwner = WinQueryWindow( hwnd, QW_OWNER, FALSE );
            WinSendMsg( hwndOwner,
                        message,
                        mp1,
                        MPFROM2SHORT( CMDSRC_MENU, SHORT2FROMMP(mp2) ) );
        }
        return MRFROMLONG( 0L );
        case WM_CHAR:
            if ( (!(CHARMSG(&message)->fs & KC_KEYUP))
                && (CHARMSG(&message)->fs & KC_VIRTUALKEY)
                && (CHARMSG(&message)->vkey == VK_ESC) )
            {
                hwndParent = WinQueryWindow( hwnd, QW_PARENT, FALSE );
                WinSetFocus( HWND_DESKTOP,
                            WinWindowFromID( hwndParent, FID_CLIENT ) );
            }
            return ((*buttonMenuFrameProc)( hwnd, message, mp1, mp2 ));
        case WM_QUERYDLGCODE:
            return (MRESULT) DLGC_MENU;
        default :
            return ((*buttonMenuFrameProc)( hwnd, message, mp1, mp2 ));
    }
}

/*****
 * FUNCTION: CubFormatFrameSubProc
 * Intercepts WM_FORMATFRAME and WM_QUERYFRAMECTLCOUNT messages from a
 * standard frame window and adds the CubButtonMenu to the format.
 *****/
MRESULT EXPENTRY CubFormatFrameSubProc( hwndFrame, message, mp1, mp2 )
HWND      hwndFrame;

```

Figure 2. Button down code (continued on page 91)



```
USHORT message;
MPARAM mp1;
MPARAM mp2;
{
    USHORT cFrameControls;
    switch(message)
    {
        case WM_QUERYFRAMECTLCOUNT:
            cFrameControls = SHORT1FROMMR( (*formatFrameProc)( hwndFrame,
                                                                message,
                                                                mp1,
                                                                mp2 ) );

            cFrameControls++;

            return MRFROMSHORT( cFrameControls );

        case WM_FORMATFRAME :
            (*formatFrameProc)( hwndFrame, message, mp1, mp2 );
            cFrameControls = FormatFrameControls( hwndFrame, mp1, mp2 );
            return MRFROMSHORT( cFrameControls );

        default :
            return ((*formatFrameProc)( hwndFrame, message, mp1, mp2 ));
    }
}

/*****
 * FUNCTION: FormatFrameControls
 *
 * Responds to a WM_FORMATFRAME message by decreasing the size of the
 * client window, in order to add a button menu to the frame window.
 * Calls WinFormatFrame and WinSetMultWindowPos.
 *****/
USHORT PASCAL FormatFrameControls( HWND hwndFrame, MPARAM mp1, MPARAM mp2 )
{
    PSWP aSwp;
    USHORT cSwp;
    HAB hab;
    HWND hwndClient;
    USHORT index;

    hwndClient = WinWindowFromID( hwndFrame, FID_CLIENT );
    hab = WinQueryAnchorBlock( hwndFrame );
    aSwp = (PSWP) PVOIDFROMMP( mp1 );
    for ( index = 0; (aSwp[index].hwnd != hwndClient); index++ );
    aSwp[(index + 1)] = aSwp[index];
    cSwp = index + 2;
    aSwp[index].cy -= buttonMenuHeight;
    if( aSwp[index].cy >= 0 )
        aSwp[(index + 1)].cy = buttonMenuHeight;
    else
    {
        aSwp[(index + 1)].cy = buttonMenuHeight + aSwp[index].cy;
        aSwp[index].cy = 0;
    }
    aSwp[(index + 1)].y = aSwp[index].y + aSwp[index].cy;
    aSwp[(index + 1)].hwnd = hwndButtonMenu;
    WinSetMultWindowPos( hab, aSwp, cSwp );
    return cSwp;
}
```

Figure 2. Button down code (continued from page 90)



handed orientation. One person wants the numbers like a telephone pad with 1-2-3 at the top, another wants the opposite.

Another important aspect about designing this touch-screen application was that keystrokes had to be minimized at every opportunity. Every time the broker has to touch a button or write on the screen, the broker's eyes would be diverted from the pit. We did not want this. So, we built in defaulted assumptions wherever possible.

When it comes to touch-screen applications, programmers have to be prepared for supporting all kinds of user-to-user requirements. They have to be flexible. That flexibility must be built right into the application.

SPEED KILLS?

Speed is always essential, so the buttons in an activity have to be grouped close together. The broker's hand must not fly all over the screen.

For example, sometimes a broker is in the act of writing out an endorsement in one window and receives an incoming order in

another window. The problem is that the "accept" button is on the far side of the screen. Although this is where the broker usually wants it, in times like these the broker also wants another "accept" button nearby to save time.

GIVE 'EM WHAT THEY WANT WHEN THEY NEED IT

You cannot expect users to be computer literate. You cannot expect them to know how to size a window or how use the title bar. If a window was expanded to take up the whole screen, the user could become confused. So we changed the windows so that the title bar is hidden unless it is specifically requested. It is embedded in the application in a turnkey way. Figure 3 shows that we concealed the frame controls.

IN CONCLUSION

Touch screen technology has opened up new areas for previously untapped automation. This technology can be successful in environments few thought possible. As you work to build the infrastructure for a better tomorrow, we hope our experiences and

coding examples may be of some assistance. Have fun!

Rolando Arquiza, senior programmer analyst, joined the CUBS team recently. His formidable background in Windows and OS/2 enable him to make significant contributions to the CUBS systems.

Jerry Bayer, senior systems analyst, contributed greatly to the development of CUBS. Bayer's expertise in OS/2 and communications has won him key roles on a number of critical projects at the CME.

Larry Gavin, manager-systems development, has been on the CUBS project since its inception. His background in trading systems covers the major Chicago exchanges including CBOE, CBOT, and CME.

Brian Norquist, senior programmer analyst, made critical contributions to the CUBS system. His extensive software engineering experience, including C++, positions him nicely as the lead developer.

Jim Zeidel, director—systems development, has also been on the project since its inception. His extensive industry background complements his responsibility over a number of trading systems used at both CBOT and CME.

```
// HIDE TITLE BAR and FRAME Controls
HWND      hwndMinMax;      /* Frame MinMaxButton Window handle */
HWND      hwndSysMenu;     /* Frame System Menu Window handle */
HWND      hwndTitleBar;    /* Frame Title Bar Window handle */
PFNWP     sizingFrameProc; /* window procedure of standard frame
                           * window as subclassed to restrict
                           * frame sizing */

BOOL      fFrameControlsHidden = FALSE;

/*****
 * FUNCTION: HideFrameControls
 *
 * Hide the title bar and associated controls
 *****/
VOID EXPENTRY HideFrameControls( HWND hwndFrame )
{
```

Figure 3. Hide title bar and frame controls (continued on page 93)



```
fFrameControlsHidden = TRUE;

hwndTitleBar = WinWindowFromID( hwndFrame, FID_TITLEBAR );
hwndSysMenu = WinWindowFromID( hwndFrame, FID_SYSMENU );
hwndMinMax = WinWindowFromID( hwndFrame, FID_MINMAX );

/* Hide by setting parent to the HWND_OBJECT window */
WinSetParent( hwndTitleBar, HWND_OBJECT, FALSE );
WinSetParent( hwndSysMenu, HWND_OBJECT, FALSE );
WinSetParent( hwndMinMax, HWND_OBJECT, FALSE );

WinSendMsg( hwndFrame,
            WM_UPDATEFRAME,
            (MPARAM) (FCF_TITLEBAR | FCF_SYSMENU | FCF_MINMAX),
            NULL );
}

/*****
 *   FUNCTION: ShowFrameControls
 *
 *   Show the title bar and associated frame controls
 *****/
VOID EXPENTRY ShowFrameControls( HWND hwndFrame )
{
    fFrameControlsHidden = FALSE;

    /* reset parent of windows to the frame window, from HWND_OBJECT */
    WinSetParent( hwndTitleBar, hwndFrame, FALSE );
    WinSetParent( hwndSysMenu, hwndFrame, FALSE );
    WinSetParent( hwndMinMax, hwndFrame, FALSE );

    WinSendMsg( hwndFrame, WM_UPDATEFRAME,
                (MPARAM) (FCF_TITLEBAR | FCF_SYSMENU | FCF_MINMAX),
                NULL );

    /* highlight the title bar assuming the window is active */
    WinSendMsg( hwndTitleBar, TBM_SETHILITE,
                MPFROMSHORT( TRUE ),
                MPFROMLONG( 0L ));

    /* force a redraw of added controls */
    WinInvalidateRect( hwndFrame, NULL, TRUE );
}

/*****
 *   FUNCTION: CubRestrictFrameSizing
 *   This function subclasses the frame window to prevent sizing when the
 *   frame controls are hidden.
 *****/
VOID EXPENTRY CubRestrictFrameSizing( HWND hwndFrame )
{
    sizingFrameProc = WinSubclassWindow( hwndFrame,
                                           (PFNWP)CubFrameSizingSubProc );
}

/*****
 *   FUNCTION: CubFrameSizingSubProc
```

Figure 3. Hide title bar and frame controls (continued on page 94)


```

*   Intercepts WM_TRACKFRAME messages from a standard frame window
*   and prevents moving and sizing when frame controls are hidden.
*   This procedure allows for disabling sizing even when the sizing
*   border is present.
*****/
MRESULT EXPENTRY CubFrameSizingSubProc( HWND hwndFrame, USHORT message,
                                         MPARAM mp1, MPARAM mp2 )
{
    switch ( message )
    {
        case WM_TRACKFRAME:
            if ( fFrameControlsHidden )
                return MRFROMSHORT( FALSE );
            else
                return ((*sizingFrameProc)( hwndFrame, message, mp1, mp2 ));

        default :
            return ((*sizingFrameProc)( hwndFrame, message, mp1, mp2 ));
    }
}

```

Figure 3. Hide title bar and frame controls (continued from page 93)

OS2TREE™

See your entire system graphically at a glance!

"What XTREE GOLD is to DOS and More!

And What Norton Commander Missed"

- Completely programmable: every function/key can be customized!
- Display your tree structure for all system and LAN disks concurrently!
- Manipulate your files, directories, entire disks and/or entire system!
- HPFS fully supported
- Pop-up list of files for any directory
- Edit/browse file(s) using any EDITOR or PROGRAM!
- Extensive FILE Search and Tree directory search capabilities make locating files and/or data easy.
- Upload/download files between mainframe and PC
- Secure delete (sensitive files wiped clean)

Intro Price: \$79.99

Professional Version: \$249.99

Limited time special \$150.00

Network Licenses Available

LEVINE COMPUTER

CONSULTING SERVICES

7640 Provincial Dr., Suite 213, McLean, VA 22102

Orders only (800) 383-9495

FAX/Inquiries/HelpLine (703) 790-1660

MAY THE SOURCE BE WITH YOU

Don't let the dark forces of ignorance defeat you. Right in this galaxy, you can tap into the source -- the free Consumer Information Catalog. It lists free and low-cost government publications on cosmic topics such as federal benefits, jobs, health, housing, educating your children, cars, and much, much more. So dispel the darkness and send for the source. Write today to Pueblo, Colorado for the free Consumer Information Catalog. Just send your name and address to:

Consumer Information Center
Department Source
Pueblo, Colorado 81009



The staff of OS/2 Developer put together this special section to guide you through the various computer-based training programs available for OS/2. The products described in this guide are based on surveys sent to vendors; they are provided as a service to you and are free to vendors. These listings do not represent an endorsement by OS/2 Developer. Although every effort has been made to ensure accuracy, we do not assume any responsibility for error or omission in this section. Please contact the companies directly for more information.

OS/2 Training Buyer's Guide

AMERICAN TRAINING INTERNATIONAL

Circle No. 130

Teach Yourself OS/2 v2.1 is a self-paced, computer-based training program that allows both new and experienced users to master OS/2 in three to four hours. It features a split-screen approach to training; the background screen simulates OS/2, while a special window provides interactive instruction to guide the user step by step through OS/2 procedures. The user reduces learning time and frustration by actually performing the skills and simultaneously seeing the effects of each keystroke. Price: \$75.00.

American Training International, 12638 Beatrice St., Los Angeles, Calif. 90066, (800) 955-5284 or (310) 823-1129, fax (310) 827-1636.

ONEONONE COMPUTER TRAINING

Circle No. 131

How to Use OS/2 version 2.x is a course on four audiocassettes of about two hours each that provides the beginning user with a hands-on orientation to the system: navigating the desktop, managing drives and directories, moving data between applications, and file editing. Data disk and quick reference guide are included. Price: \$195.00.

OneOnOne Computer Training, 2055 Army Trail Rd., Dept. MF1,

Addison, Ill. 60101, (800) 424-8668 or (708) 628-0500, fax (708) 628-0550.

USA TRAINING

Circle No. 132

Working with OS/2 version 2.1 is a four- to six-hour video course that focuses on features and functions of OS/2, including Workplace Shell's graphical interface. This course can help novices learn basic concepts and show experienced users how OS/2 differs from DOS and Windows. Price: \$99.00.

USA Training, 13515 Dulles Technology Dr., Herndon, Va. 22071, (800) USA-2002 or (703) 713-3813, fax (703) 713-0065.

VIAGRAFIX

Circle No. 133

Getting Started with OS/2 gives an overview of OS/2, including multitasking, graphical interfacing, on-line information, Presentation Manager, and boot manager. It covers OS/2 fdisk and Workplace Shell use, migration, installing and running various applications, and understanding basic commands. Fifty-four minutes. Price: \$39.95.

Productivity with OS/2 explains the path, the system editor, menus, folders, objects, associating programs with datafile objects, customizing programs, editing icons, adding parameters, manipulating objects, making objects

DON'T MISS OUT



on a single issue of OS/2

OS/2 Developer offers tips and techniques for more efficient computing for the advanced software developers who have recognized the need for 32-bit power.

TO ORDER

For new orders & customer service:

1-800-WANT-OS2

(1-800-926-8672) OR

708-647-5960

FAX (415) 905-2233

Written subscription orders:

OS/2 Developer • P O Box 1079
Skokie, IL 60076-8079

Subscription rates for 6 issues:

- ☐ U.S. — \$39.95
- ☐ Canada — \$55.95
- ☐ International surface mail
- ☐ International air mail — \$69.95
- ☐ Check inclosed

Charge my:

- ☐ Visa ☐ MasterCard ☐ AmEx

Card # _____

Exp. Date _____

Signature _____

Name _____

Company _____

Address _____

City _____

State/Zip _____

Make checks payable to Miller Freeman, Inc. Foreign orders must be in U.S. funds. Please allow eight weeks for subscriptions to begin.

HOUSAD

from templates, exchanging data between sessions, printing, and running multiple applications. Eighty-seven minutes. Price: \$49.95.

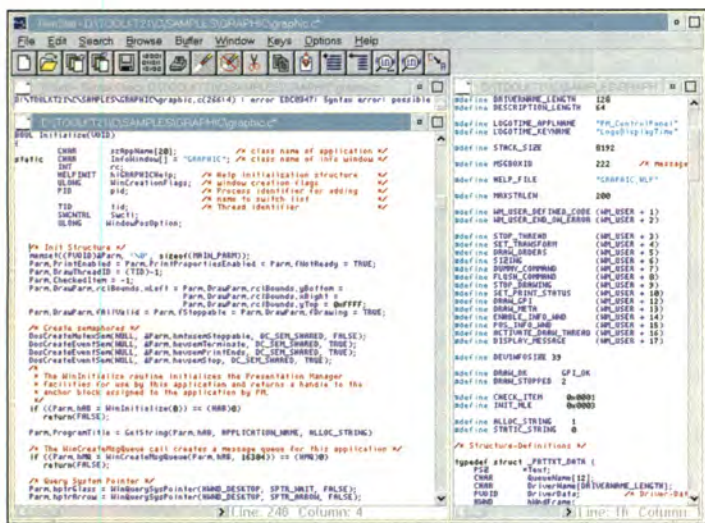
OS/2 for your LAN defines local area network and gives an overview of LAN server version 3.0. It addresses sharing resources, controlling LAN access, planning a network, software, hardware, and compatibility questions. It highlights installing the server, requester, and the DOS LAN requester. It explains configuration concepts of the communication environment. Eighty-two minutes. Price: \$59.95.

Optimizing OS/2 for HOT Performance looks at CONFIG.SYS, the path and libpath commands, setting devices, modifying setup, and setting application environment parameters. This video covers file settings, memory management, and processor performance. It presents how to observe multitasking levels, reduce task switching operations, improve file system, and speed up your system. Sixty-eight minutes. Price: \$39.95.

Each video comes with a learning disk. Videos can be purchased individually or as a set; all four videos are \$159.95.

ViaGrafix, 5 S. Vann St., Pryor, Okla. 74361, (800) 842-4723 or (918) 825-6700, fax (918) 825-6744.

POWER UP WITH THE RIMSTAR PROGRAMMER'S EDITOR NEW VERSION 2.1



If you're looking for a fully configurable, professional OS/2 PM programmer's editor to replace your current text-mode editor – we have what you have been looking for!

No hassle changing editors

We know changing editors can be a major hassle. You don't want to relearn a new set of keystrokes no matter how good the product. Well, you don't have to – we have Brief, CUA, Epsilon, Multi-Edit, SlickEdit, PWB, and Borland IDE keyboard mappings waiting for you. If you're not using one of these, let us know and we will create the mapping you need.

Features designed to increase productivity

It has all the features you have come to expect, plus special features designed to anticipate your needs and increase your productivity. Features like a 'C' source browser that eliminates those time consuming searches for functions, global variables, typedefs, and macros by providing you with instant positioning to both definitions and references. Using our powerful 'C' macro language you can customize your programming environment to suit your individual needs.

RimStar Technology, Inc.

91 Halls Mill Road
Newfields, NH 03856-0938
Voice: (603) 778-2500
Fax: (603) 778-2408

Price \$299.00

Plus Shipping & Handling.

To order call 1-800-746-7007

60 day money-back guarantee.

Also available for Windows and Windows NT

Circle Reader Service Number 43

Partial Feature List

- ✓ Complete ANSI 'C' macro language
- ✓ SyntaColor™—syntax highlighting
- ✓ Smart C/C++ indenting & brace matching
- ✓ Bookmarks
- ✓ Unlimited undo and redo
- ✓ Timed auto save
- ✓ Access PDK help for function under cursor
- ✓ Multi-threaded for no waiting
- ✓ Compile and jump to errors
- ✓ Import/export to system clipboard
- ✓ Keystroke record/playback
- ✓ Block indent/outdent
- ✓ Hex editing
- ✓ Customizable menus
- ✓ Column, line and block selection, search and replace
- ✓ Integrates with Workframe/2
- ✓ Source browser for 'C'
- ✓ Template editing
- ✓ Multi-buffer regular expression search and replace
- ✓ Support for version control
- ✓ Complete on-line help
- ✓ Save and restore state between sessions
- ✓ OS/2 2.x 32 bit PM Multi-document interface

"My copy of Brief has been permanently retired. Keep up the good work!" -A.L.

All products and company names are trademarks or registered trademarks of their respective holders. RimStar and SyntaColor are trademarks of RimStar Technology, Inc.

© 1994 RimStar Technology Inc.

OWL is the cross-platform framework with the future built in

You're a programmer. So when it comes to choosing a framework upon which to build your applications, you know that a good object-oriented design is critical to development success. Therein lies the difference between Borland's ObjectWindows® Libraries (OWL) and Microsoft's Macro-based Foundation Classes (MFC). OWL is very object-oriented, MFC isn't.

MFC is not designed with the safety that exception handling and templates provide. And it's not architected for the cross-platform future.

OWL's superior architecture, inherent safety, and cross-platform advantage will make your development efforts more successful.

OWL has over 200 object-oriented advantages

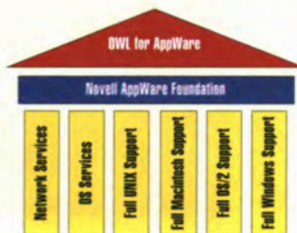
Borland's leadership in object-oriented technology shines throughout OWL. Unlike MFC, OWL is not just a thin veneer over Windows API calls. OWL delivers real benefits by increasing the reusability, reliability, and flexibility of your code.

What does this mean in coding reality? Suppose you want to make a toolbar. If you want a black and white toolbar with 16x15 pixel bitmaps using default actions, OWL and MFC are about the same. But don't be fooled. Suppose you need larger buttons? Suppose you need an edit field in your toolbar? Or you need to add new functionality to a status bar? OWL gives you immediate object-oriented flexibility. MFC doesn't.

And this is just one example. With over 200 classes, encapsulating everything ranging from display contexts to print preview windows, OWL will keep you on the object-oriented fast track.

Be safe

If your code is your livelihood, you want to make sure your code is safe. No questions. OWL is designed from the ground up with exception handling built in. So if something



"OWL isn't just a simple wrapper around Windows API.

It is a well-designed class library in the fullest sense of the word.

Anything that MFC can do, OWL can do better."

1994—Software Development Magazine

unexpected happens, you have the power of C++ backing you up. You can easily add specialized error handling. You can easily correct problems. All without inelegant return value checks and other "out of date" error approaches. And if you don't write your own handlers, OWL's built-in handlers will take care of the problems for you, with full stack unwinding and destructor calling, for safe cleanup.

Get the cross-platform advantage

ObjectWindows 2.0 is the foundation for ObjectWindows for AppWare, a true cross-platform solution for Windows, Windows NT, Chicago, OS/2,* Macintosh, and UNIX. So you develop your code once, and deploy it on whatever platforms you need. While at the same time leveraging the capabilities of each target platform.

OWL for AppWare is being built upon two proven, successful products, ObjectWindows and Novell's AppWare Foundation. Join the ObjectWindows for AppWare Early Experience Program and get started with cross-platform development today.

Borland gives you easy access to the information you need

Download the full 60-page OWL vs. MFC

comparison paper from one of these on-line services:

- Borland Forums on CompuServe, BIX, GENie
- Borland Bulletin Board (408) 431-5096
- Through Internet via anonymous ftp at <ftp.borland.com>

Join the OWL for AppWare Early Experience Program

- Call now (408) 431-1507

Subscribe to Borland's free electronic newsletter and get a regular update on the hottest programming tips and tricks.

- For more information, send an electronic mail to tech-info@borland.com

Buy Borland C++ Today and put OWL 2.0 to work for you. Call: **1-800-645-4559, ext. 8930**



Borland
The Upsizing Company

CALL NOW 1-800-645-4559, EXT. 8930